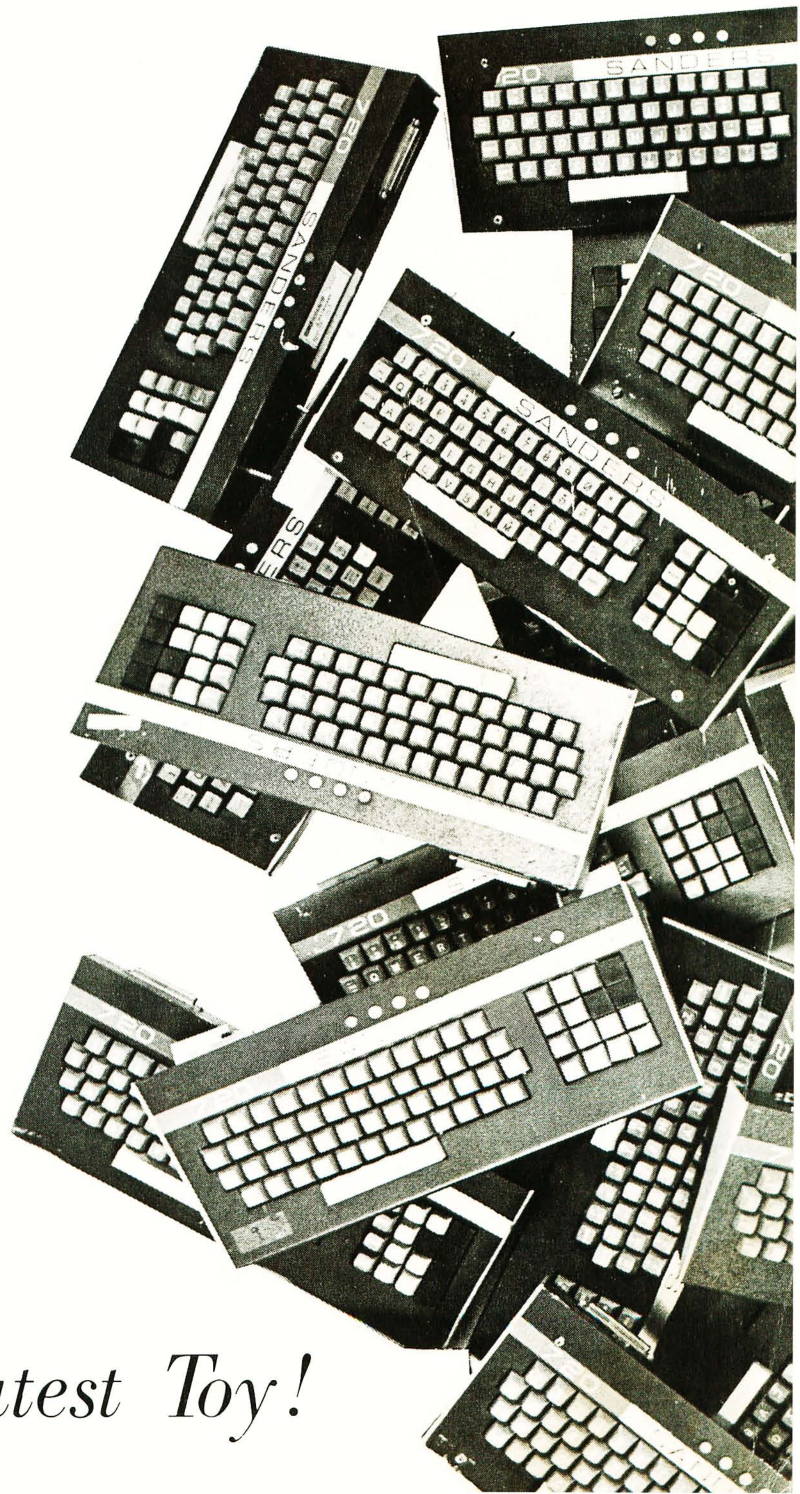
ISSUE #1

SEPTEMBER 1975

BYTE

$1.50

the small systems journal



Which Microprocessor for you?

Cassette Interface — Your key to inexpensive bulk memory

Assembling Your Assembler

Can YOU use these SURPLUS KEYBOARDS? (You bet you can!)

COMPUTERS -

the World's Greatest Toy!

The MODULAR MICROS from MARTIN RESEARCH

Here's why the new *MIKE 2* and *MIKE 3* are the best values in microcomputers today!

8008 OR 8080

Martin Research has solved the problem bothering many potential micro users .... whether to go with the economical 8008 microprocessor, or step up to the powerful 8080. Our carefully designed bus structure allows either processor to be used in the same system!

The *MIKE 3* comes with an 8080 CPU board, complete with crystal-controlled system timing. The *MIKE 2* is based on the 8008. To upgrade from an 8008 to an 8080, the user unplugs the 8008 CPU board and plugs in the 8080 CPU. Then he unplugs the 8008 MONITOR PROM, and plugs in the 8080 MONITOR PROM, so that the system recognizes the 8080 instruction set. That's about it!

If the user has invested in slow memory chips, compatible with the 8008 but too slow for the 8080 running at full speed, he will have to make the 8080 wait for memory access—an optional feature on our boards. Better still, a 4K RAM board can be purchased from Martin Research with fast RAM chips, capable of 8080 speeds, at a cost no more that you might expect to pay for much slower devices.

In short, the *MIKE 2* user can feel confident in developing his 8008 system with expanded memory and other features, knowing that his *MIKE 2* can be upgraded to a *MIKE 3*—an 8080 system—in the future.

EASE OF PROGRAMMING

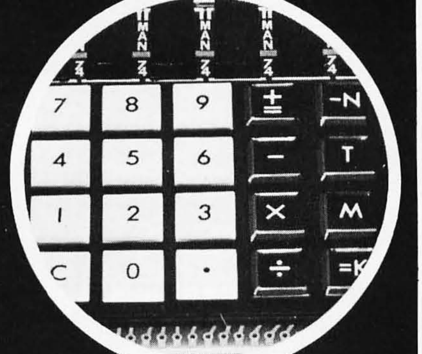
Instructions and data are entered simply by punching the 20-pad keyboard. Information, in convenient octal format, appears automatically on the seven-segment display. This is a pleasant contrast to the cumbersome microcomputers which require the user to handle all information bit-by-bit, with a confusing array of twenty-odd toggle switches and over thirty red lights!

A powerful MONITOR program is included with each microcomputer, stored permanently in PROM memory. The MONITOR continuously scans the keyboard, programming the computer as keys are depressed.

Say the user wishes to enter the number *135* (octal for an 8008 OUTPUT 16 instruction). He types *1,* and the right-hand three digits read *001.* Then he presses *3,* and the digits say *013.* Finally he punches the *5,* and the display reads *135.* Notice how the MONITOR program

(Continued in column 3.)

The MIKE *Family*



Introducing the family of modular micros from Martin Research!

Choose either the economical 8008 processor, or the powerful 8080. Either CPU is compatible with our advanced bus structure! Plus, a convenient monitor program, in PROM memory, allows you to enter instructions with the ease of a handheld calculator. Six large digits display data in octal format.

Modularity makes for easy expansion. First quality parts throughout. Professionally made PC boards with plated holes, soldermask protection. 8080 CPU board features versatile interrupt structure, multiprocessing capability. Easy interfacing to input and output ports.

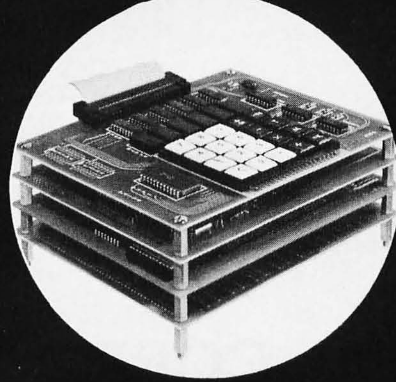
MIKE 303A: CPU board with 8080, keyboard/display board, PROM/RAM board monitor PROM (256 bytes of RAM), breadboard, hardware, and instructions: $395.00 kit, $495.00 assembled and tested.

MIKE 203A: CPU board with 8008, keyboard/display, PROM/RAM, breadboard, hardware, and instructions: $270.00 kit, $345.00 A&T.

MIKE 3-5 or 2-5: 4K RAM board with 450 ns static RAM: $165.00 kit, $190.00 A&T.

FREE CATALOG!

Kits: US & Canada only.
Master Charge accepted.
OEMs: write for quantity prices.



MARTIN RESEARCH
Microcomputer Design
1825 S. Halsted St.
Chicago, IL 60608
(312) 829-6932

shifts each digit left automatically as a new digit is entered! The value on the display is also entered into an internal CPU register, ready for the next operation. Simply by pressing the *write* key, for example, the user loads *135* into memory.

The MONITOR program also allows the user to step through memory, one location at a time (starting anywhere), to check his programming. Plus, the Swap Register Option allows use of the interrupt capabilities of the microprocessor: the MONITOR saves internal register status upon receipt of an interrupt request; when the interrupt routine ends, the main program continues right where it left off.

We invite the reader to compare the programmability of the *MIKE* family of microcomputers to others on the market. Notice that some are sold, as basic units, *without any memory capacity at all.* This means they simply cannot be programmed, until you purchase a memory board as an "accessory." Even then, adding RAM falls far short of a convenient, permanent MONITOR program stored in PROM. Instead, you have to enter your frequently-used subroutines by hand, each and every time you turn the power on.

EASY I/O INTERFACE

The *MIKE* family bus structure has been designed to permit easy addition of input and output ports. A hardware interface to the system generally needs only two chips—one strobe decoder, and one latching device (for output ports) or three-state driving device (for inputs). A new I/O board can be plugged in anywhere on the bus; in fact, all the boards in the micro could be swapped around in any position, without affecting operation. I/O addresses are easy to modify by reconnecting the leads to the strobe decoder (full instructions are provided); this is in marked contrast to the clumsy input multiplexer approach sometimes used.

POWER & HOUSING

The micros described to the left are complete except for a cabinet of your own design, and a power supply. The basic micros require +5 V, 1.4 A, and −9 V, 100 MA. The 4K RAM board requires 5 V, 1 A. A supply providing these voltages, and ±12 V also, will be ready soon.

OPTIONS

A number of useful micro accessories are scheduled for announcement. *In addition,* the *MIKE 3* and *MIKE 2* may be purchased in configurations ranging from unpopulated cards to complete systems. For details, phone, write, or check the reader service card.

COMPUTER EXPERIMENTER SUPPLIES

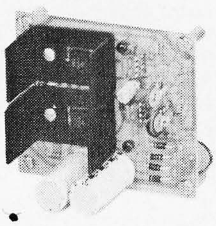
FACTORY FRESH—PRIME QUALITY
PERFORMANCE GUARANTEED

MICROPROCESSORS AND MEMORY

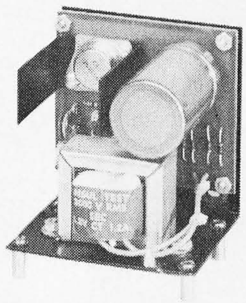
8008	$ 35.00	Commercial Grade—up to 35°C.
8080	135.00	These units are factory fresh, full spec devices.
2102	3.50	These units are factory fresh, full spec devices.
2102-2	4.50	These units are factory fresh, full spec devices.

COMPUTER GRADE REGULATED POWER SUPPLIES

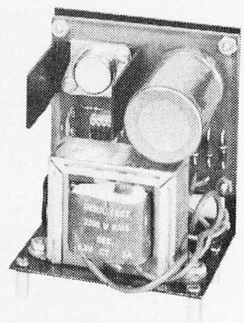
All units are short-circuit proof, fold back current limited and with over-voltage crowbar protection.



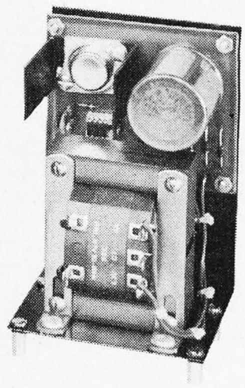
MD-15
±15 Volt at 200MA
Dual Tracking
$30.00



MD-5-1
+5 Volt at 1 Amp
$24.50



MD-5-3
+5 Volt at 3 Amp
$34.50



MD-5-6
+5 Volt at 6 Amp
$44.50

MICRO COMPUTER SUPPLY COMBINATIONS

For the 8008
MD-08—+5 volt at 6 amp, -12, -9 at 200 ma $75.00
For the 8080
MD-80—+5 volt at 6 amp, ±12v at 200 ma ...$75.00
For the Fairchild F-8
MD-8—+5 volt at 6 amp, +12 v at 200 ma ...$65.00
For the M6800
MD-5—+5 volt at 6 amp $44.50

All units are short circuit proof, fold-back current limited and with over voltage crowbar protection.

TTL INTEGRATED CIRCUITS

All devices are factory fresh, full spec units.

7400	.23
7404	.25
7442	.60
7447	.95
7448	.95
7475	.60
7490	.60
7493	.60
74125	.55
74126	.55
74192	1.10
74193	1.10

All Prices Subject to Change Without Notice
Minimum Order $10.00
Add $1.00 to Cover Postage and Handling
Send Check or Money Order (No C.O.D.) To:
N. J. Residents Add 5% Sales Tax

GUARANTEE
Most devices shipped within 24 hours. If not shippable within 2 weeks payment refunded. Performance guaranteed on all units for 30 days. Defective parts replaced at no charge. NOTICE: This warranty applies **only** to parts that have not been soldered. You must use sockets for your incoming inspection tests.

MICRO DIGITAL CORP.

BOX 413, EDISON, NJ 08817 • (201) 549-2699

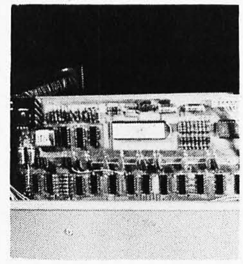
In the Queue

BYTE #1
SEPTEMBER 1975

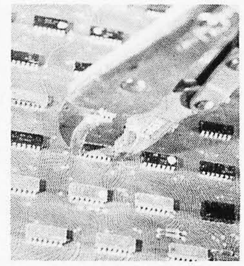
Foreground

Background

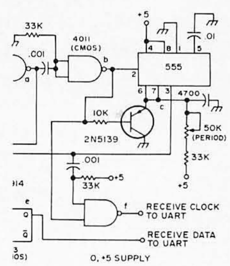
Nucleus



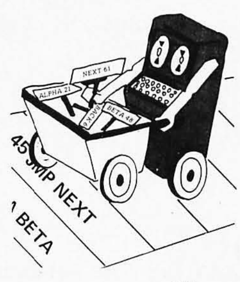
p. 10



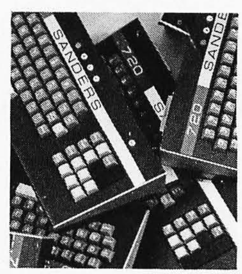
p. 20



p. 22



p. 50



p. 62

BYTE magazine is published monthly by Green Publishing, Inc., Peterborough, New Hampshire 03458. Subscription rates are $12 for one year worldwide. Two years, $22. Three years, $30. Second class postage application pending at Peterborough, New Hampshire 03458 and at additional mailing offices. Phone: 603-924-3873. *Address editorial correspondence to Editor, BYTE, Box 378, Belmont MA 02178. From inception to press in seven weeks — surely a magazine creation record. Guinness please take notice.*

Carl Helmers:

What is BYTE?

"It could not have been long before some wizard of verbal magic figured out that a group of little bits must constitute a mouth watering BYTE."

"... the term byte has become part of the lexicon."

For the hardware person, "the fun is in the building."

This is the first issue of a new publication – BYTE – a monthly compendium of information for the owners and users of the new microcomputer systems becoming widely available at moderate cost. To formal and informal students of computer science, the choice of the name BYTE is quite appropriate. For a large number of applications of this new technology of inexpensive computers, character string and text, data (basic unit, one byte) is an important consideration. Bytes are the units of data manipulated by many of the small computer systems designed by readers – or assembled using one of a number of kit products now on the market.

The most common definition of a byte is that of a unit of information containing 8 bits. This unit of information can at any time represent one of $2^8 = 256$ possible things – for instance, one of the ASCII or EBCDIC character codes, one of the integers from 0 to 255, a signed integer from -128 to +127, etc. The origin of the term "byte" lies in IBM's documentation and terminology for the extremely successful System 360 series. The folk tale has it that IBM needed a more "personalized" (i.e. unique) term for the old standby of earlier generation computers, the "character". The term had to be less tied to a specific type of data such as character codes – and had to take on a generic meaning as "unit of storage". With that functional specification for the required term, it could not have been long before some wizard of verbal magic figured out that a group of little bits must constitute a mouth watering byte.

With the term's widespread use in the computer field due to IBM's benign influence, the term byte has become part of the lexicon. The fundamental significance of a byte as a unit of information makes BYTE an appropriate name for the publication. BYTE is your unit of information on the state of the art of small computer systems for individual persons, clubs and classroom groups. Each month you will find information ranging from computer club announcements to manufacturers' advertisements, from technical details of hardware and software to humorous articles and editorial opinions.

The Home Brew Computing Trilogy

The story of computing is a story composed of several elements. A good way to look at the story is as a trilogy of interrelated themes . . .

HARDWARE

SOFTWARE

APPLICATIONS . . .

You need the hardware before you can progress through the first gate of a system. A virgin computer is useless so you add some software to fill it out. And the whole point of the exercise – in many but not all cases – is to come up with some interesting and exotic applications.

The technical content of BYTE is roughly divided into the trilogy of hardware, software and applications. Each component of the trilogy is like a facet of a brilliant gem – the home brew computer applied to personal uses. The trilogy is not confined to home brew computers alone of course.

In the personal computing field as in any endeavor there are people who will have foremost in their minds any one of these three topics to the exclusion of the other two. For instance some of the people I know are interested in software – and pretty exclusively software. They'll tend to concentrate on software as much as possible and try to get a minimal amount of hardware sufficient to experiment with software. A person with a primary interest in software will oftentimes be the person who purchases a kit computer because the kit minimizes the amount of hardware knowledge the person is required to have.

An example of another kind of person – in terms of isolated characteristics – is the hardware kind of person. Here the emphasis of the work with home brew computing is on putting together the hardware, designing the hardware, making things that quote "work" unquote. A "hardware person" in many cases may not do much else – but he or she certainly will accomplish the design goal. This is the person who builds

— (the first) editorial

up a computer system to the state where it might even be able to do a bootstrap off tape — then drops the system and decides to build a better one. The fun here is in the building, not in the using and programming.

Then in this description of possible ways of approaching the home brew hobby there is the applications person. This person's attitude is somewhat a synthesis of the other two types. The applications person is typically interested in getting a particular program up and running. So a Space War freak would spend a good portion of available time getting the hardware and software needed to play space war. A LIFE addict would spend a fair amount of time getting the hardware and software for the game of LIFE — and fooling around with LIFE patterns. And a person who enjoys other computer games — using the game as a goal — spends much time assembling a hardware/software system for the game. A person interested in toy robots would have a combined hardware/software problem of coordinating and controlling movement — this home roboteer must design the mechanical details, design a control algorithm — and if sophisticated fun is required, must design a pattern recognition input device and algorithm for interpreting scenes. A model railroader requiring a computer controlled layout again has this applications model — computer controlled yard and main line switches — and faces a choice of possible hardware and software components needed to make the application.

Now, aspects of the trilogy exist in any particular person who experiments with the computer systems. A common combination is for the application to drive the hardware and software choices. Then there is the person who builds the hardware first — getting caught up in the "neatness" of a logical construction in the same way that a mathematician goes off the deep end with a neat theorem. If you're starting as such a hardware hacker, you may come to the point where you say "Hmmm — I've built the hardware, so now what do I do with it?" Here the applications are following the design of the computer. It's the same way with many kinds of programming . . . the software is an exploration of the possibilities of the hardware. As a software hacker you might turn to the pure logic of programming — writing and trying out routines for things ranging from augmentations of the instruction set to file managers, to games simple and sophisticated. Or you may just fool around with programming with no specific end in mind in terms of applications, for the sole purpose of seeing what you can do with the machine.

As a home brew computer software experimenter, you'll find an emotional kinship with the people who take part in the automotive hobbies. What does a "performance" automobile buff do with the machine — once all the optional improvements and features have been added underneath the personalized paint job? The auto nut takes his car out to the local drag strip or other test track and opens up the throttle to see what the engine and drive train will do in terms of speed and acceleration (proper spelling: exhilaration). Well, for computer experimenters there is a logical drag strip in every computer — an instruction set waiting to be explored. You exercise this logical drag strip by seeing what you can do in the invention of neat little (and not so little) programs to do useless (equivalent of real drag strip) performance tests in artificial circumstances — or really useful tasks (equivalent to normal transportation functions of autos). I haven't yet figured out what the computer equivalent of an air scoop hood is — or the equivalent of the weird mechanical contrivances I often see on the derriere of "muscle" cars.

You might even be the type of person who wants to do a certain programming technique just for the sake of programming — you say to yourself: "OK — I want to write a BLURPTRAN language compiler and code generator, so what hardware do I need to do it?" As such a person, you would then choose a computer system in packaged and/or self-designed form such that it would fit the compiler writing goal.

In truth many will find it best to seek a sort of

"A virgin computer is useless so you add some software to fill it out . . ."

Sophisticated fun requires sophisticated thought and hard work . . .

"Hmmm — I've built the hardware, so now what do I do with it?"

"Well, for computer experimenters there is a logical drag strip in every computer — an instruction set waiting to be explored . . ."

interactive — balanced — relationship between the three aspects of the computer trilogy. At any given time, almost anyone has some aspects of all three combined within his own philosophy of home brew computing.

BYTE — the magazine — addresses this mixture that occurs in various people by providing articles permuting and combining these areas.

In this first issue *hardware* articles include "Deciphering Mystery Keyboards," an article on recycling used ICs, and Don Lancaster's article on serial output interfaces. Articles on *software* include a description of the assembler concept by Dan Fylstra. *Applications* are found in the first segment of LIFE Line. This application includes information on programming techniques as well as suggestions regarding required hardware.

So here in the first issue you find an example of the mixtures of these factors which go into home brew computing. This mixture of aspects is a guiding theme of the current and following issues of BYTE — one of the key editorial goals is to cover a complete range of ideas spanning this triumvirate of concepts.

The Impossible Dream

or, "Wouldn't it be neat to have a computer all one's own without being as rich as Croesus?"

The art of home brew computing has come a long way in the past few years. To paraphrase the science fiction author Robert Heinlein, "When it's time to do home brew computing, people do home brew computing." That time has come today, with the advances in memory and processor technology inherent in large scale integration. The present devices are not as good as the "Thorsen Memory Tubes" in Heinlein's *Door Into Summer* — but it's getting almost to the point where a basement tinkerer can put together a manufacturable robotic device and plant the economic acorn which will grow into an industrial oak tree.

My own first exposure to the idea of home brew computing was about eight years ago when I was attending high school in rural New Jersey. A ham (radio amateur variety) friend of mine at that time was attempting to get a surplus RCA computer card rack into operation as his own conception of a home processor. I didn't know enough at the time even to ask an intelligent question about its design. The thing was a monstrous 3-level card rack with a heavy wire wrap back plane and transistor logic with integration to the level of modular cards. I don't think this friend of mine ever got his processor working to any significant extent — but the impression was made: "Wouldn't it be neat to have a computer all one's own without being as rich as Croesus?" I filed away the thought of a home brew computer as an "impossible" dream at that time — how could I afford a computer if I could barely afford a beat up old Hallicrafters SX-99 receiver and flea power ham transmitter? That did not stop me from having fun with computers — it merely caused a redirection of attention to the use of computers financed by agencies other than myself . . . for a while.

The while lasted several years as I bootstrapped myself through college with FORTRAN, COBOL, PL/1, BAL and a bit of financial aid from a private foundation. Along about 1972 when I started reading about the LSI computers being designed by Intel — the 8008 and 4004 — I began to revive that old dream of "having a personal computer." Here was a single IC chip — the 8008 — which would give me a real stored program machine at reasonable ("reasonable" = under $1000) cost. After attending an Intel seminar in 1972, I resolved that I would actually build an 8008 computer.

The resolution was a long time being turned into reality — I did not actually begin design and construction until January 1974. I took my time for numerous reasons . . . among them being the fact that I had to learn something about the way hardware works, had to equip a laboratory of sorts, got this bug about self-publishing the results along the way*, and so on . . . I finally got an 8008 computer which would execute instructions in the middle of the summer of 1974 — and to quote one of the subscribers to my self-published series of articles, "I learned a lot . . .," just as he did. So much for the personal involvement — I've built a kluge of sorts — the software hacker's first attempt at hardware — and have learned quite a bit as a result. You — the reader of BYTE — can go that route or use a much easier route — there are several manufacturers of kit products advertising in this magazine. And you'll find a magazine full of helpful information which I didn't have.

. . . CARL

*I got the idea of self-publishing from a pamphlet put out by SOL III Publications, Farmington, Maine — which has since been turned into a book entitled *The Shoestring Publisher's Guide* — full of useful materials on publishing by individuals and small organizations.

World's Most Inexpensive BASIC Language System

$995

Limit: one per customer.
OFFER expires September 15, 1975.



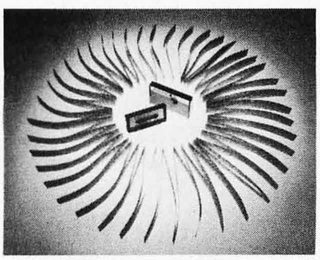
Altair 8800 Computer Kit

Two 4,096 word Memory Boards (kit)

Your choice of Interface Boards (kit)

Altair 8K BASIC Language

Altair 8K BASIC Language. This language was chosen for the Altair Computer because of its versatility and power and because it is easy to use (comes with complete documentation). Altair 8K BASIC has many features not normally found in BASIC language including an OUT statement and corresponding INPut function that allows the user to control low speed devices (machine control without assembly language). Leaves 1750 words in 8K machine for programming and storage.



NOTE: Altair BASIC comes in either paper tape or cassette tape. Specify when ordering.

Interface Board Options. The *Parallel Interface Board* is used to connect the Altair 8800 to external devices that send and receive parallel signals. Many line printers require a Parallel Interface Board. The *RS232 Serial Board* is used to connect the Altair 8800 to external devices that send and receive RS232 serial signals. Most computer terminals require an RS232 Serial Interface Board. The *TTY Serial Interface Board* is used to connect the Altair 8800 to an ASR-33 or KSR-33 teletype (20 milliamp current loop). The *TTL Serial Interface Board* is for custom interfacing. The *Audio Cassette Interface Board* is used to connect the Altair 8800 to any cassette tape recorder. It works by changing the electrical signals from the computer to audio tones. It can be used to store unlimited amounts of information coming out of the computer and it can be used to put information back into the computer.

PRICES:

Altair Computer kit with complete assembly instructions	**$439**
Assembled and tested Altair Computer	**$621**
1,024 Word Memory Board	**$97 kit and $139 assembled**
4,096 Word Memory Board	**$264 kit and $338 assembled**
Full Parallel Interface Board	**$92 kit and $114 assembled**
Serial Interface Board (RS232)	**$119 kit and $138 assembled**
Serial Interface Board (TTL or TTY—teletype)	**$124 kit and $146 assembled**
Audio Cassette Interface Board	**$128 kit and $174 assembled**
4K BASIC language (when purchased with Altair, 4,096 words of memory and Interface Board)	**$60**
8K BASIC language (when purchased with Altair, two 4,096 word memory boards and Interface Board)	**$75**
COMTER II	**$780 kit**
Teletype ASR-33	**$1500 (assembled only)**

Input Output Devices. The Comter II Computer Terminal has a full alpha-numeric keyboard and a highly-readable 32-character display. It has its own internal memory of 256 characters and complete cursor control. Also has its own built-in audio cassette interface that allows you to connect the COMTER II to any tape recorder for both storing data from the computer and feeding it into the computer. Requires an RS232 Interface board.

The Standard ASR-33 Teletype prints 10 characters per second. It has a built-in paper tape reader and punch. Has standard 120 day Teletype warranty. Requires a Serial TTY Interface board.

NOTE: The Altair 8800 can be connected to any number of input/output devices other than the ones listed above.

MITS

"Creative Electronics"

6328 Linn, N.E., Albuquerque, NM 87108 505/265-7553

MAIL THIS COUPON TODAY

☐ Enclosed is check for $________

☐ BankAmericard #________ ☐ or Master Charge #________

* ☐ $995 BASIC System Special with following Interface Board: ☐ Parallel ☐ Serial RS232 ☐ Serial TTY ☐ Serial TTL ☐ Audio Cassette

☐ Altair 8800 ☐ Kit ☐ Assembled ☐ Options (list on separate sheet)
Include $8 for postage and handling

☐ **Please send free Altair System Catalog**

NAME ________

ADDRESS ________

CITY ________ STATE & ZIP ________

Credit Card Expiration date ________

MITS/6328 Linn, NE, Albuquerque, NM 87108 505/265-7553

Warranty: 90 days on parts for kits and 90 days on parts and labor for assembled units. Prices, specifications and delivery subject to change.

from the Publisher . . .

how BYTE started

Two series of events came together and triggered BYTE. One was the surprising response I received from the readers of 73 Magazine (amateur radio) every time I published an article involving computers. Being a curious person I decided to learn more about them, only to find my way blocked by formidable obstacles. The more I tried to dig into the subject the more I found that there was a need for information that was not being satisfied.

The other event was the success of 73, with more subscriptions and advertising calling for some sort of computerization of the drudgery – the billing, record keeping, reader's service, indexing, and such. I knew what I wanted done and had a good idea of what I had to spend to accomplish this, so I started talking to computer salesmen . . . only to find that I wasn't even able to read their literature, much less have even a vague idea of what they were saying.

Some deep well of obstinacy within me fought back and refused to let me *throw* a dart to pick out the computer system I needed. I felt that as a businessman running a good sized small business and as the editor and publisher of an electronics magazine, I damned well should be able to come to grips with the salesmen and pick out a computer system on some sort of rational basis. But the more I tried to get information, the more I realized that it was going to be very hard to get.

Between my professional need to understand computers and my amateur interest in the subject I found myself subscribing to one newsletter after another . . . talking at exhaustive length with computer savvy 73 readers . . . reading books . . . and wearing computer salesmen out. I discovered an interesting thing – few of the hardware chaps could talk software – and vice versa. Further, neither could talk much about applications.

There ought to be a magazine covering the whole thing, thought I. A magazine which would help the neophite to grapple with programming languages . . . would permit the beginner to build microcomputers and peripherals . . . would provide a dialog for the more sophisticated to communicate as well. How about a publication which would cover all aspects of small computer systems?

As the computer hobby newsletters arrived I looked them over. Some were very well done, some pretty juvenile. One chap was doing a splendid job . . . designing his own hardware . . . developing software . . . plus writing and publishing a monthly magazine on the subject just about single handed. This was Carl Helmers and his ECS Journal, which was in its fifth issue, having just started in January (this at the time being May). I got together with Carl and explained my idea and suggested that it was time to get a good professional magazine going in the field, one which would help computer hobbyists get the information they needed and which might thus encourage manufacturers to come out with more hardware for the growing body.

Carl had been building up his circulation to ECS gradually, with it being about 300 in May. We figured to go all out and run off 1000 copies of the first issue of

continued on page 96

BYTE staff

EDITOR
Carl T. Helmers Jr.

ASSISTANT PUBLISHER
Judith Havey

ASSOCIATE EDITORS
Dan Fylstra
Chris Ryland

CONTRIBUTING EDITORS
Hal Chamberlin
Don Lancaster

EDITORS
John Burnett
Susan G. Philbrick

PRODUCTION MANAGER
Lynn Panciera-Fraser

ART DEPARTMENT
Nancy Estle
Neal Kandel
Peri Mahoney
Bob Sawyer

PRINTING
Biff Mahoney

PHOTOGRAPHY
Bill Heydolph

TYPESETTING
Barbara Latti
Marge McCarthy

ADVERTISING
Bill Edwards
Nancy Cluff

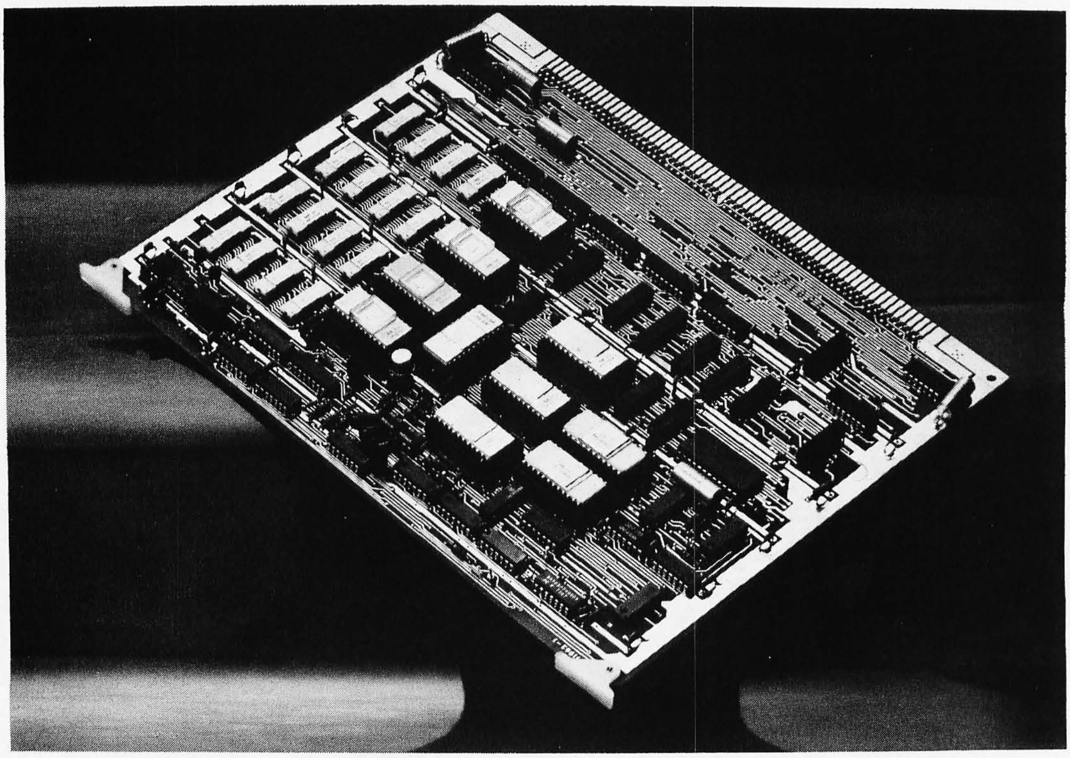
MARKETING
David Lodge

CIRCULATION
Susan Chandler
Dorothy Gibson
Pearl Lahey

INVENTORY CONTROL
Marshall Raymond

DRAFTING
Bill Morello

Which Microprocessor



National Semiconductor's IMP-8 and IMP-16 computers were originally offered as completely populated subsystem cards such as the one pictured here.

by
Hal Chamberlin
Box 295
Cary NC 27511

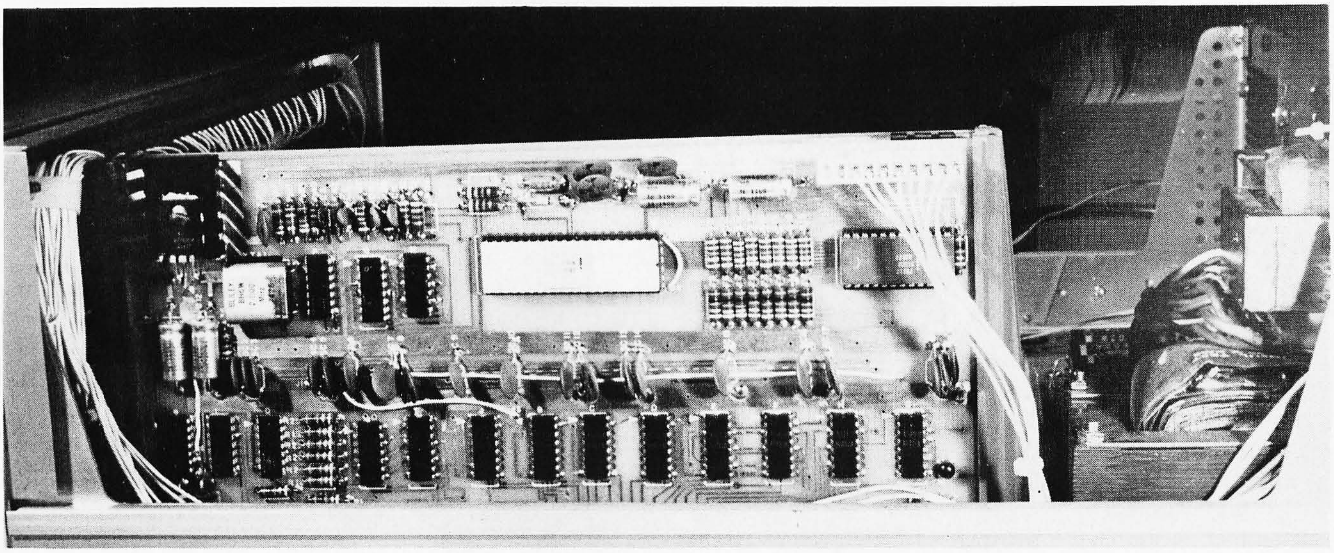
At this time there are three microprocessor chips or chip sets readily available to the hobbyist: the 8008, the 8080, and the IMP-16. The first two were pioneered by Intel and the last is a National Semiconductor invention. Chips and/or kits utilizing each of the three microprocessors are available from at least two sources catering to hobbyists as of this writing. This level of availability and popularity is not even approached by other microprocessors, therefore this discussion is being confined to these three.

Comparing computers is like comparing people: the conclusions depend on the application, the circumstances, and personal preference. The comparisons made will be based on use of the microprocessor as a general purpose computer. For our purposes a general purpose computer is one which has read-write memory for the bulk of its storage, which is expected to run a variety of programs, and for which the end use is the development and execution of programs written by the user. General purpose computers are also expected to be able to control a variety of input-output equipment. Instruction sets will be compared on the basis of assembly language programming. Speed will be compared on the basis of the time necessary for the machine to complete a non-trivial task. Complexity will be compared on the basis of ease of understanding microprocessor operation as well as the sheer number of parts required to implement a system. Finally, cost will be compared on the basis of minimum systems capable of assembling programs for themselves given the existence of suitable I/O devices.

Before getting into comparisons, we will take a brief look at the leading

Reprinted from The Computer Hobbyist, *Box 295, Cary NC 27511.*

for You?



The MITS Altair 8800 is one package in which you can purchase an 8080 based system.

features of each microprocessor. Then the comparisons will be made in each performance area elaborating on individual features as necessary.

Comparing computers is like comparing people; the conclusions depend on the application, the circumstances, and personal preference.

The Intel 8008 Processor

The 8008 was the first microprocessor to be introduced and the first to be available to the hobbyist. It has an 8 bit instruction and accumulator length. There are essentially only two memory addressing modes: immediate, and zero displacement indexed. Subroutine and branch addresses are full length absolute, allowing branching anywhere with one instruction. Subroutine return addresses are saved on an internal 8 level stack which puts a 7 deep restriction on subroutine nesting. Much of the instruction set power is derived from the six additional 8 bit index registers which may count, save, or address memory. The maximum directly addressable memory is 16k bytes; in addition, 8 input and 24 output devices may be directly addressed with a one byte instruction. CPU speed is modest ranging from 20 microseconds for a register operation to 32 microseconds for a memory operation to 44 microseconds for a jump or call. A selected chip, the 8008-1, reduces these times to 12.5, 20 and 27.5 microseconds respectively. A single level of interrupt is provided but external hardware is necessary for complete status saving during interrupts. Interfacing the chip to the rest of the system is fairly involved and requires from 20 to 70 TTL packages depending on the system performance desired. The lower figure will barely function while the higher one includes a console, complete interrupt system, and dynamic memory interface with direct memory access capability. Most of the interfacing complexity can be blamed on overzealous designers trying to make-do with an 18 lead package. Present cost to the experimenter ranges from $40 to $80 with the "dash one" version bringing roughly 50% more.

The Intel 8080 Processor

The 8080 is Intel's sequel to the 8008. Basically it has more of everything. The instruction set contains all of the 8008 instructions making it upward compatible at the assembly language level. Major additions to the instruction set include direct load and store of the accumulator, double precision (16 bits) add and increment for address calculation, and a pushdown stack of indefinite length in memory thus allowing unrestricted subroutine nesting. Addressable memory has been increased to 64k bytes and addressable I/O devices have been increased to 256 inputs and 256 outputs at the expense of 2 byte I/O instructions. Execution speed has been considerably improved also. Register operations take 2 microseconds, memory operations require about 3.5 microseconds, and subroutine calls consume 8.5 microseconds. Interrupts work the same way as on the 8008 but everything required for complete status saving is provided as well as an interrupt enable/disable flag. Interfacing an 8080 is generally regarded as being simpler than interfacing an 8008. There is only a slight

The 8080 is Intel's sequel to the 8008. Basically it has more of everything . . .

improvement in the minimum system, about 15 chips, but a full-bore system may be cut in half to 35 chips. The 40 lead package allows a separate 16 bit address bus and 8 bit data bus, as well as simplified timing and control. Present cost to the hobbyist is about $160.

The National IMP-16

The IMP-16 is one of the older microprocessors and for a long time the only one with a 16 bit wordlength. The programmer is supplied with four 16 bit accumulators and a 16 word stack. The instruction set is typical of many 16 bit minicomputers, and in many ways resembles that of a NOVA. Four general address modes are provided, base page direct, program counter relative, and indexed using either accumulator 2 or accumulator 3. In addition, LOAD, STORE, JUMP, and CALL can be indirect addressed using any of the addressing modes to get to the address pointer. Two memory modification instructions are provided, ISZ (Increment memory, Skip if Zero), and DSZ which allows much counting and indexing to be done in memory freeing the registers for arithmetic. The stack is used for subroutine return addresses but can also be used for saving registers and status. A unique feature is the availability of an extended instruction set chip which provides automatic multiply, divide, double word add and subtract, and byte manipulation. The CPU can address 64k words but this should be held to 32k if the byte instructions are used. The I/O instructions can also address 64k devices. Another unique feature is that several bits of input and output are provided by the microprocessor itself making communication with a teletype possible without any interface at all. Speed is good ranging from 4.2 microseconds for a register operation to 7 microseconds for a memory operation. A multiply takes about 160 microseconds which is still considerably faster than a software routine would be. The IMP-16 provides two priority levels of interrupt and all of the hardware necessary for complete status save/restore. Interfacing is conceptually simple and requires 25 to 50 packages depending on system sophistication. Part of this number is due simply to the fact that 16 bits are to be handled rather than 8. The microprocessor is in the form of five 24 lead packages which for the most part are simply wired in parallel. The extended instruction set resides in a sixth package. Present cost of the standard chip set is about $160. The extended instruction set chip is available only from National at this time for $80.

The IMP-16 is the first of the 16-bit micros . . .

A good instruction set should be well organized . . .

Comparisons — Instruction Sets

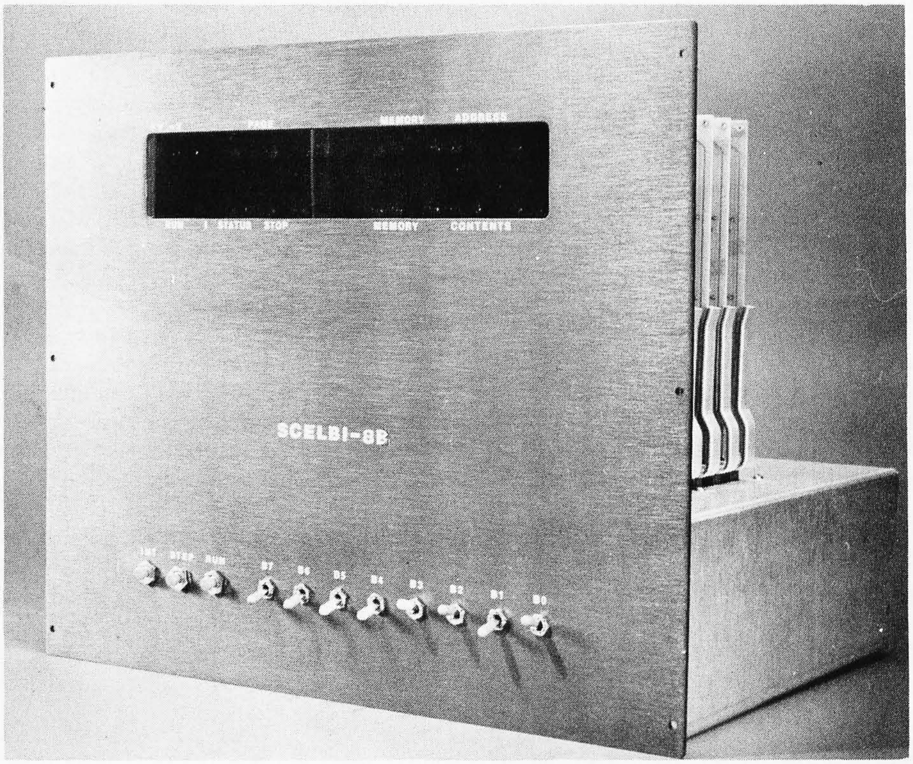
One of the most important performance areas of a microprocessor is the instruction set. A good instruction set should be well organized so that it is easy to learn, powerful so that complex routines can be coded with a small number of instructions, memory efficient so that complex routines require only small amounts of memory, and time efficient so that only a small number of memory cycles is necessary to complete a task. In addition, performance should be equally high on both character oriented tasks and numerically oriented tasks.

Instruction set organization is best on the 8080 closely followed by the 8008 with the IMP-16 being somewhat disorganized. Consequently, the beginner will find the 8008/8080 the easiest to learn. Experience has shown that beginners prefer simple instruction sets and that they retain a certain "fondness" for their first machine long after they have graduated into much more sophisticated endeavors. The experienced programmer however should experience little difficulty keeping the little quirks, distinctions, and special cases straight when working with the IMP-16.

Instruction set power is best on the IMP-16, followed by the 8080 with the 8008 a distant third. Based on actual experience, it may require as few as one half as many IMP-16 instructions to program a task as 8008 instructions. The 8080 falls about midway between the extremes. There are many reasons why the IMP-16 is superior. Memory addressing is much more flexible due to the four addressing modes and indirect addressing capability. An additional advantage is that the arithmetic word length is the same as the address length. Since the return addresses are put on a stack in all three machines, multiple entrypoint subroutines are easy but the IMP-16 also allows multiple return points (return to CALL+1 on error, CALL+2 otherwise, etc.) with no additional instructions. The 8080 is a big improvement over the 8008 because registers may be saved on the stack when they are used by a subroutine and then restored unaltered upon return. This allows subroutines to be called as needed without regard to which registers they may destroy. Note, however, that this capability may be added to the 8008 quite simply. The direct load and store instructions of the 8080 reduce the number of lines of code in a program.

Memory efficiency of the instruction set is best on the IMP-16 but is closely followed by the 8080. The 8008 is not as bad as might be presumed but is definitely

Instruction set organization and memory efficiency are usually conflicting requirements.



Scelbi Computer Consulting Inc. is one of a number of companies who take the Intel 8008 computer, package it into a system design, and sell the result as a system. The photograph supplied by Scelbi with a press release shows you the result of assembling their new "SCELBI-8B" version of the 8008. As a supplier to the computer enthusiast market from the start, Scelbi has done a very credible job of assembling a true system product as opposed to a bare-bones CPU which merely blinks its lights after assembly.

inferior. In terms of numbers, the 8080 may require 10 to 15 percent more memory bits and the 8008 20 to 40 percent more. Note that these figures are based on optimized programs written by experienced programmers. The spread can be much greater with inexperienced programmers or hastily written programs. It is also interesting to note that instruction set organization and memory efficiency are usually conflicting requirements. This is because many of the lesser used possible operation combinations have been culled from a memory efficient set in order to reduce the number of bits required to encode the instruction. Implied operands are also utilized in order to free up bits for other uses. Experienced programmers are able to plan ahead and avoid having these restrictions become restrictive. The 8008 and 8080 are as good as they are because many of the instructions are a single word (8 bits) long whereas the minimum instruction length in the IMP-16 is 16 bits. This is somewhat offset by the three word (24 bit) instructions of the 8008 and 8080 which in most cases would only require 16 bits in the IMP-16. A fringe benefit of high memory efficiency is that the shorter programs will load faster regardless of the loading method.

Time efficiency is by far the best on the IMP-16 with the 8008 a distant second and the 8080 a slightly poorer third. On a classic minicomputer, a machine cycle was the same as a memory cycle in most cases. As a result, a time efficient instruction set meant a faster machine without faster hardware. Microcomputers on the other hand may have very few of their machine cycles being memory cycles. As a result, time efficiency may have little relation to actual machine speed but does represent the potential speed with an optimized CPU. Time efficiency can be important in multiprocessor systems with a shared memory where more memory cycles increase the probability that a CPU will have to await its turn. The IMP-16 has a high time efficiency mainly because twice as much data is fetched in each memory cycle. Further improvement is due to the instruction set power, requiring fewer instructions to be fetched. The 8080 has poorer time efficiency than the 8008 mainly because the stack is in memory. A subroutine call, for example, requires 5 memory cycles, 3 to fetch the instructions and two to stack the return address.

Historically some minicomputers were better at handling character oriented tasks and others were well adapted to number crunching tasks. Microcomputers are no exception. Most micros have been optimized for character handling because of expected high usage in terminals and the 8008 and the 8080 belong to this class. The IMP-16 on the other hand is much better at numerically oriented tasks and was aimed more toward machine tool control and industrial monitoring. Interestingly, use of the extended instruction set on the IMP-16 greatly improves both character handling and arithmetic capability.

Most micros have been optimized for character handling because of expected high usage in terminals.

How About Running System Software?

One performance area of interest to hobbyists is the suitability of a machine for running a BASIC system. The IMP-16 and the 8080 are about equal in their ability to compile BASIC quickly but the IMP-16 without the extended instruction set may

Speed in a hobby computer system can be a two-edged sword.

execute BASIC twice as fast. This is due mainly to the all floating point arithmetic that BASIC requires. The extended instruction set may double the speed again if a lot of multiplies and divides are done. The 8008 can of course run BASIC also but compile and execution speeds are likely to be one tenth of the 8080.

One other property of an instruction set is the ease with which it may be assembled, either by hand or with an assembler program. In this respect, the 8008 comes out on top with the 8080 next and the IMP-16 last. Use of the mnemonics and format recommended by the manufacturer is assumed in making this comparison. 8008 code is easy to hand assemble because the octal notation used corresponds to the various fields in the instruction word. Assemblers for 8008 code can also be quite simple because instructions require at most one operand and very few instruction formats exist. Further simplification results from all addresses being absolute and all mnemonics being three characters. 8008 assemblers run on an 8008 can be as small as 2.5k bytes for a limited implementation but 4k bytes is more realistic when providing an easy to use assembler. If the hexadecimal notation recommended by Intel is used with the 8080, hand assembly is definitely more difficult. The assembler also has a tougher time with the two operand format and other niceties defined for the 8080. The Intel version of the 8080 assembler requires 8k bytes but it should be noted that it provides macro capability. Hand coding and assembling for the IMP-16 is harder yet due mainly to relative addressing considerations and a wider variety of instruction formats. National's version of the assembler requires 4k words and can produce relocatable object code and handle external symbols.

Is Speed Useful?

Speed in a hobby computer system can be a two-edged sword. A high speed microprocessor requires higher speed in other system components such as memory in order to realize its higher speed. An 8008 for example can run at full speed with memories as slow as 3 microseconds access but the 8080 will have to wait on memories slower than 520 nanoseconds and the IMP-16 requires 420 nanoseconds. If the ready line is used on the 8008 and 8080 to permit the use of slower memories, the wait will be in increments of whole machine cycles which is 4 microseconds on the 8008 and 500 nanoseconds on the 8080. Thus if memory is a tad slow, one cycle will be added to each three cycle memory access sequence slowing the system down an average of 25 percent to 30 percent. The IMP-16 does not have a ready line, rather the user stretches one of the clock periods in a cycle long enough to permit memory access. This scheme has the advantage that the stretch can set to the exact amount needed. The higher time efficiency of the IMP-16 instruction set will greatly reduce the performance impact of a slow memory as compared to the 8080.

How Complex is the Interface?

The ranking on interface complexity is 8080 (least), IMP-16, and 8008. The comparison is based on sophisticated general purpose implementations having a complete I/O interrupt facility, software console using ASCII I/O, and a generalized input/output memory bus allowing simultaneous direct memory access without affecting the CPU. The ranking is the same whether parts count or conceptual complexity is being considered. The difference between the 8080 and the IMP-16 is primarily due to the wider word of the IMP-16 and less confusing discussion of chip interfacing in the Intel manual. The 8008 is just plain difficult to understand and interface correctly but once that is done, either by the user, a magazine, or a manufacturer, the system should operate just as well.

Software?

Software support is often a big issue among industrial users of microprocessors. Unfortunately, the majority of the software they are fighting over is unavailable to the hobbyist because of high prices. It is not unusual for a program such as an assembler to cost as much as a handful of microprocessors. 8008 and 8080 users can look to Scelbi and MITS for some software at reasonable prices even if they did not purchase their machines from these sources. National has an excellent body of software for the IMP-16 but the package price is $200 for object tapes and source listings. The hobbyist *will* have to depend on himself, kit manufacturers, and publications for most of his software in the near future. Ultimately, the level of software support will be directly proportional to the popularity of the microprocessor.

And Finally . . .

There are a number of other performance areas that are only of minor interest to hobbyists. Although there is a large spread in the maximum memory size, all three machines are likely to have ample addressing capability for the hobbyist's memory budget. The same may be said relative to addressable I/O devices. Power supply voltages and power consumption are also usually of minor importance. All of the microprocessors can be successfully operated from standard +15, +5 and -15 system supply voltages using simple, fail-safe, zener regulators. Package size and pinout are unlikely to be factors in hobbyist use.

This brings us to a comparison of overall system cost. First, the spread in chip cost is roughly from $50 to $150, so the spread in system cost would be $100 at most. An 8008 requires more interfacing circuitry however which reduces the spread somewhat. After enough memory to do assemblies or run BASIC and a few I/O devices are added, the $75 difference left may be small compared to the total investment. Nevertheless, an 8008 system will be the least expensive followed by an 8080 system closely followed thereafter by an IMP-16.

Which microprocessor for you? The answer still depends on the application, circumstances, and personal preference, but hopefully the decision can be made with more authority after reading this article.

Last Chance –



DON'T MISS THE BOAT!

This is the last opportunity to become a CHARTER SUBSCRIBER to BYTE . . . at the special Charter rate of $10 per year.

Perhaps you've held off to see the first issue . . . perhaps you've just seen a friend's copy of BYTE . . . whatever the reason . . . send in your $10 and be sure to get your next issue of BYTE hot off the presses.

If you are reading a pass-along copy, just mark the subscription order that you'd like your subscription to start with issue number 1.

Name____________________

Address____________________

City__________ State______ Zip______

☐ BILL ME

☐ Start with #1 BYTE

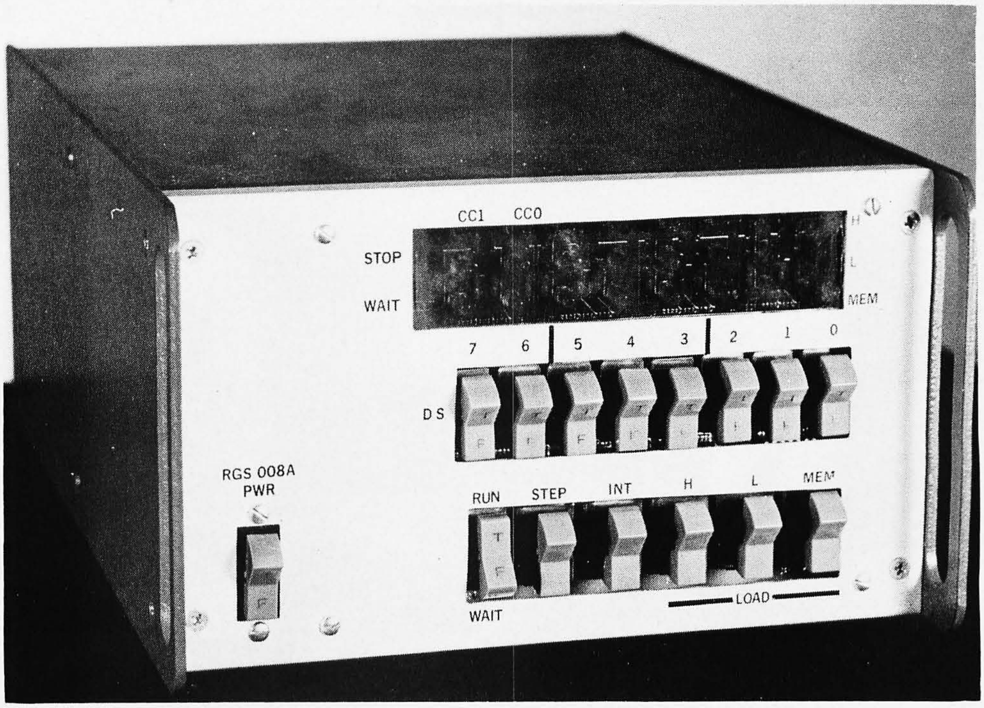
☐ Check for $10 enclosed

☐ Bill BankAmericard or MasterCharge #

☐ Start with #2 BYTE

My occupation____________________

BYTE Peterborough NH 03458



The RGS 008A Microcomputer Kit

Review

by

James Hogenson

Box 295

Halstad MN 56548

The RGS 008A microcomputer is a general purpose machine based on the Intel 8008 CPU chip. The 008A uses a minimum of 1k of random access memory and a bus type of I/O system capable of handling up to 256 peripheral devices.

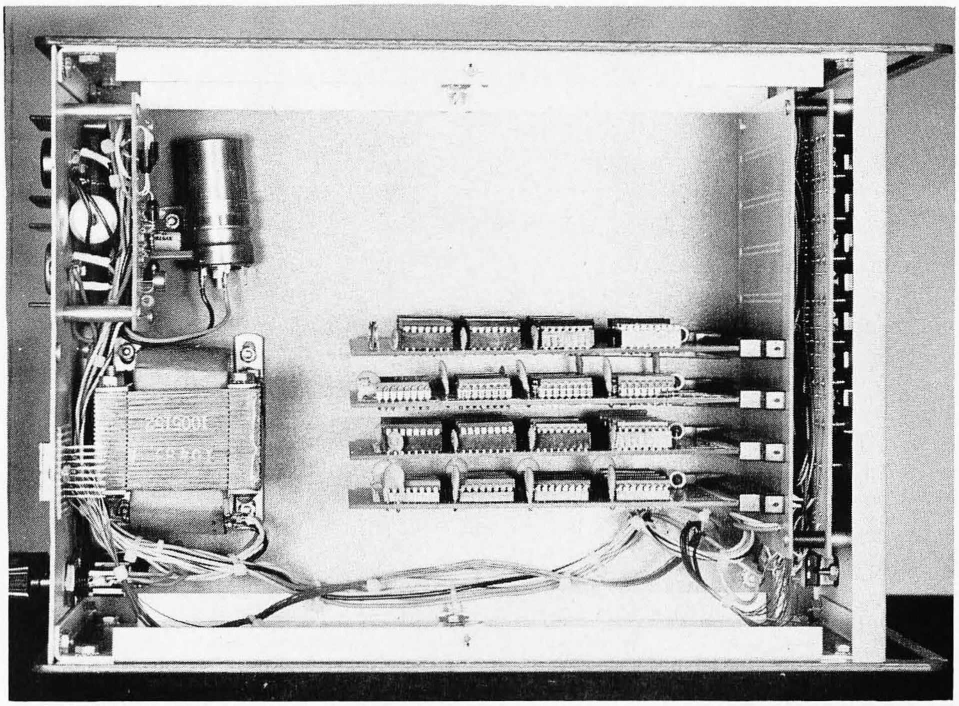
The basic kit consists of 6 printed circuit boards and all components necessary to build the CPU, 1k of memory, the control panel and the power supply. Molex pins or sockets, edge connectors, backplane, front panel switches and LEDs, and a power transformer are included. A cabinet, some hardware, line cord and fuses are not included. The kit sells for $375, making this one of the least expensive kits on the market.

Physical Construction

The control switches are mounted directly on the control panel PC board, together with the LED indicators. A printed circuit backplane board is connected to the back side of the control panel. Once the control panel and backplane are wired together, anything plugged into an edge connector on the backplane is automatically connected to the system. The control panel and backplane measure 4.8" x 6.75". The double-sided plug-in boards measure 4" x 6" with 72-pin edge connectors, and have plated through holes. The CPU with all control circuitry included is fitted onto only two of these plug-in boards.

Microcomputers, microcomputers and more microcomputers! The number of options you have for a homebrew system is expanding at a rapid rate - with the corresponding difficulty in picking and choosing among the options. BYTE has an answer to this problem – reviews of kits and equipment from various manufacturers and advertisers in the magazine. The idea of a review is to give the user's evaluation of the product - and in the process aid you in the choice of equipment for your own homebrew system. Here is BYTE's first review of an 8008 product – the RGS 008A microcomputer kit (made by RGS Electronics, 3650 Charles St., Suite K, Santa Clara CA 95050).

James Hogenson provides this review of the RGS product, based upon his own experiences assembling and utilizing the computer. Jim built the kit early this year as part of his high school Science Fair project activities designing an oscilloscope CRT display. I think you'll find Jim's account to be a useful source of information on the RGS product. **. . . CARL**



Is an RGS 008A the computer for your system?

Here is one RGS 008A's owner's evaluation.

The control panel holds three 8-bit binary LED displays. Two of the displays indicate the memory address of the date or instruction presently being operated upon or executed. The third display shows the contents of the memory location indicated by the memory address displays.

The two LEDs not used in the upper memory address are used to indicate the second and third bytes in a two and three byte instruction. Two additional LEDs indicate the stopped and waiting state of the CPU.

The only switches used on the control panel are a common 8-bit binary switch register and 6 control switches. The six control switches and their functions are:

Memory Load. A momentary switch which loads the data presently on the 8-bit switch register into the memory location specified by the memory address register.

Load L. A momentary switch which loads the address set on the 8-bit switch register into the lower memory address register.

Load H. A momentary switch which loads the address on the switch register into the upper memory address register. (The three load switches will operate only when the CPU is in the stopped state.)

Interrupt. A momentary switch for entering interrupt instructions.

Step. A momentary switch for single stepping through a program. The run/wait switch must be set on wait to single step a program.

Run/Wait. A toggle switch which is used to set the CPU in either a run or a wait state. During the run state, the CPU operates normally. During the wait state, the CPU does nothing unless being single stepped.

The 008A is designed to be panel mounted. The entire system is contained on plug-in boards mounted on the back of the control panel. The entire unit may then be mounted very neatly; however, a nice improvement would be for the manufacturer to offer a cabinet to mount the unit in.

The power supply included in the kit will provide enough power to allow for moderate expansion of the system. If necessary, auxiliary power supplies can be purchased from the kit manufacturer. The power supply included in the kit will produce -5 V at 5 Amps and -12 V at 1 Amp. The -9 V for the 8008 is derived from the -12 V.

The only type of peripheral interface presently offered for the 008A is a parallel TTL-compatible interface. The parallel interface boards are also 4" x 6" boards which may be plugged directly into the backplane if desired. Up to 256 interface cards may be used. The parallel interface kit sells for $43.75.

A 008A cassette tape adapter is offered for $100. The FSK adapter's data rate is 300 bits per second. The popular 40-pin UART chip is used for parallel-to-serial and serial-to-parallel conversion. The cassette adapter kit includes a parallel interface kit for interfacing with the computer. Only a cassete deck with an auxiliary output or earphone plug and an optional remote start/stop plug is required.

An ASCII keyboard is also available from RGS. This keyboard and the cassette tape adapter are intended to interface with the RGS 008A microcomputer, so adaptation would be a problem if you have some other make of computer. RGS says that a serial interface is being developed for use with teletype, as well as a TV typewriter.

Highlights

The 008A uses fewer components, compared to other 8008 systems I have seen. The 008A computer structure is basically similar to other 8008 systems, with one exception: The input/output structure is unusual for an 8008 system. Instead of the more common port

system capable of handling only 8 input and 24 output devices, the 008A uses a bus system capable of handling 256 bi-directional data input/output channels. RGS uses several of the 8008 chip's I/O channels instructions to set up this bus I/O System.

There are four input/ output instructions in the 008A's "instruction set:" Select, Control, Input, and Output. To select a peripheral, the device number is loaded into the accumulator and a select instruction is executed. This enables the peripheral device. If necessary, a control instruction is available. Upon its execution, the accumulator data is made available at the output and the control strobe line is pulsed. Once the device has been selected and initialized, data transfer using the input and output instructions is executed as it is in the port system.

Each peripheral interface board is jumper-wired for its own device number. When the device number of a particular unit is selected, data transfer between the unit and the CPU is enabled until another device number is selected. When a control instruction is executed, the accumulator data is placed on the I/O bus. The interface transfers the data from the bus to the interface output. The control strobe line is then pulsed. An output instruction also causes the accumulator data to be placed on the I/O bus. The interface transfers the data to its output. This time the output strobe line is pulsed. On an input instruction, the interface transfers data from its input to the I/O bus. The CPU then transfers the data from the bus to the accumulator. The input strobe line is also pulsed.

The bus system has three big advantages. The obvious one is the capability of handling up to 256 devices. The second advantage is that each of the 256 channels will perform both input and output. The third advantage is the ability to send control instruction, thus eliminating the need to use an adjacent I/O channel for device control.

Ready and interrupt line expansions may also be brought out through the peripheral interface boards.

Assembly

Most of the circuitry is completed by soldering the components on the various PC boards. Although most of the PC boards are double sided, components need only be soldered on one side since all holes are plated through.

Edge connectors are provided for each plug-in board. These connectors must be soldered on the backplane. Some 68 wires must be soldered between the backplane and the control panel. The backplane and control panel are fastened back to back using spacers on long machine screws.

The control panel, CPU boards, memory boards, and parallel interface boards are double sided PC boards with plated through holes. These PC boards seem to be of good quality. The backplane board and the power supply board, however, are single sided and not quite as good. Extreme caution must be used as the foil patterns on these single sided boards will lift with very little excessive heat applied.

Only about half the power supply circuitry is included on the power supply PC board. A power transistor and a voltage regulator, both in a T0-3 case, must be mounted in a heatsink (supplied in kit) on a chassis which must be provided by the builder. The power transformer, fuses, filter capacitors and power switch are also mounted on the builder supplied chassis. All of these components must be point to point wired. Wire must be supplied by the builder.

Documentation

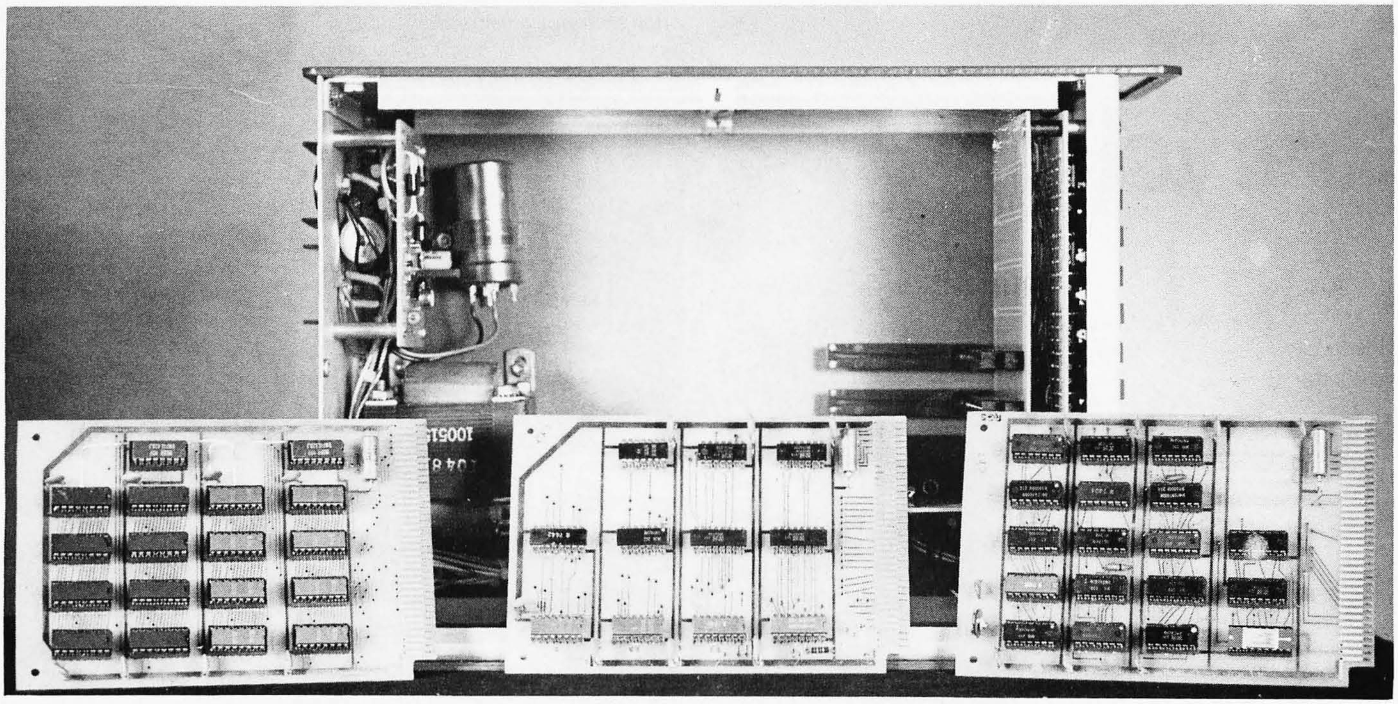
The manuals and instructions provided for the 008A fall short. Anyone planning to write up any documentation in the future should take note of this and use the criticism constructively.

The assembly instructions provided for the 008A are brief. The author of the documentation spent very little time on clarifications. Someone who has had considerable experience in both electronic engineering and computer science would have very few problems with the 008A. However, if a product is made with the intention of selling it to everyone, the documentation should be written to be understood by everyone. This includes people who are not experts in hardware or software.

Much of the construction has to be done by following the component placement diagrams and circuit diagrams. A diagram of the foil pattern line-up for double sided boards should have been included in the 008A documentation. For easier reading, component placement diagrams should be black components on a grey foil pattern diagram rather than black on black.

The 008A documentation provides only a brief definition of each software instruction. No information describing the circuit operation of the CPU is provided. A brief description of circuit operation is provided for the parallel interface kit and cassette adapter kit. Only a small amount of information is provided on the use and operation of the computer.

Although the 008A documentation does include complete circuit diagrams with IC pinouts, absolutely no troubleshooting information is included. Logic circuit troubleshooting can be followed on the circuit diagram after you have checked wiring and component placement, but test points and voltages should be given on the power supply. Test points and waveforms should be given for devices such as a cassette tape adapter. Even in the



logic circuitry, a few major test points and the specifications wouldn't hurt anyone.

This author obtained his troubleshooting information via long distance calls to California. The cause of several problems was narrowed down to about three faulty components which RGS cheerfully replaced free of charge.

A nice addition to the owners manual for those who may assemble a 008A in the future would be a series of test routines to verify all operations of the computer. These routines should state specifically, step by step, what is to be done and the results of each step. When starting up a computer for the first time, it is sometimes difficult to determine whether a problem is in the hardware or the software, especially if you are not familiar with the machine in the first place.

Software

The only difference in the 008A instruction set as compared to other 8008 based systems is the input/output instruction format as mentioned before. Initial program loading is through the control panel. Startup is achieved by setting the restart instruction on the 8-bit binary switch register and depressing the interrupt switch. As in other 8008 systems, the first 64 words of memory are divided into 8 program start-up locations.

Any single-byte instruction will be accepted by the CPU at any time when the interrupt switch is depressed. Therefore, if you get your computer into a loop with no exit, set a halt or return instruction and depress the interrupt switch to get out.

A software exchange program is slowly getting underway for the 008A. Each user is asked to submit a program to RGS. The programs are compiled as they come in, then sent out to each 008A manual holder. As of yet, exchange has been slow. Ray Stevens, owner and designer, expects the exchange program to pick up somewhat since the cassette tape adapter, ASCII keyboard, and some software for each have recently become available. The available software (sent out to owners of the units) deals with program loading and storage necessary in the development of further programs.

Conclusion

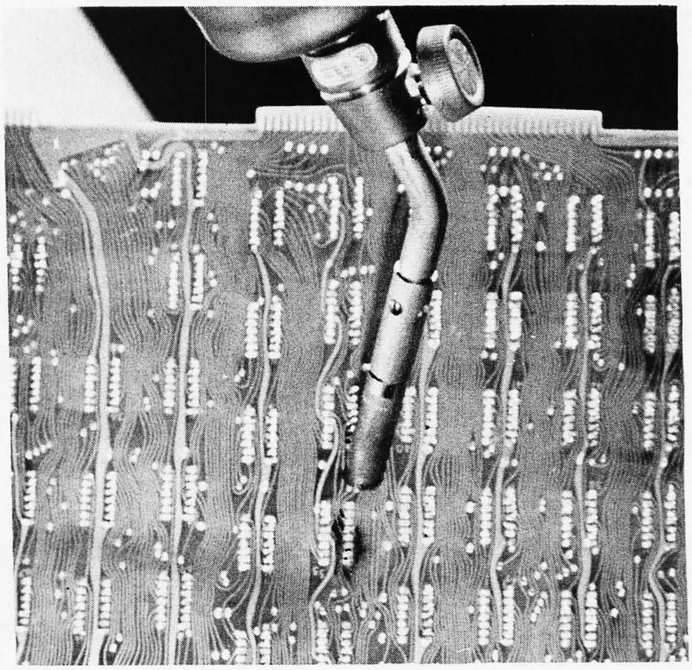
A generalized statement summarizing the 008A scene would have to be this: The 008A is one of the most economical systems on the market, offering a nearly complete kit and software exchange program. RGS holds good potential for becoming a healthy segment in the growing world of avocational computing. But there is some work yet to be done.

In the meantime, if you are inexperienced in either electronics or computer science, you could run into difficulties in assembling a 008A. Those of us who already own a 008A microcomputer should work on building up the software exchange program to a point of excellence. In the interest of obtaining a revised, updated, improved and rewritten owners manual, let me suggest that all 008A owners send RGS a letter listing changes and improvements they would like to see.

The 008A is one of the most economical systems on the market, offering a nearly complete kit and software exchange program . . . But there is some work yet to be done.

Recycling Used ICs

Sweep the blow torch over the IC's pins—one complete sweep once or twice a second.



by
Carl Mikkelsen
35 Brookline St., No. 5
Cambridge MA 02139

The surplus market is saturated with used printed circuit boards from early computer systems which offer a very inexpensive per chip source of ICs. Used boards typically contain 50-200 chips of small scale or medium scale integration, usually with many simple two input gates and four bit data registers. Common part numbers include 7400, 7402, 7404, 7408, 74126, 74174, 74175, etc. Through careful shopping, I have found boards with large numbers of multiplexors such as 74151, 74153, and even scratch pad registers — 7489. After removing chips from the boards and eliminating any non-functional units, cost per chip is from 3 to 8 cents, resulting in an overall cost of about one fourth to one tenth of the individual chip cost through other surplus outlets.

Removing chips from boards offers advantages over purchasing chips surplus which makes them attractive for reasons other than price. Primarily, the companies which originally built the boards used top-quality, fully spec'ed components. All chips have already been tested, and most have already served in equipment.

Given that you've found a serendipity of well soldered chips, it's necessary to unsolder them without either burning them or cracking their cases. Desoldering individual leads can be done, but usually the chip is made unnecessarily hot by the prolonged application of heat. Also, pulling each lead out separately results in bent, often broken leads. Devices are available which will heat all 14 or 16 pins of a small IC, but again a long time is needed to melt the solder since the total amount of energy available is limited to a small soldering pencil heating element. Most available boards are two sided and four layer boards aren't uncommon. Multi-layered boards make the required amount of energy even higher.

When a board is built, the ICs are positioned in place with all other components, and the board is soldered by a three step process.

1. The underside is washed by hot, bubbling, liquid flux.
2. The clean board is passed over a small fountain of solder, so that the board just touches it.
3. After cooling, the board is immersed in FREON gas to remove any remaining flux.

As you can see, the board is subjected to high temperatures during the soldering phase, which takes around 5-10 seconds.

The blow torch method of IC removal duplicates conditions during board soldering by heating all pins simultaneously; removing the IC is a single step.

Equipment Needed

To use this technique, you will need:

A torch. Non-oxygenated propane and acetylene gas has been used.

Clamps or a vise to hold the board fairly rigid during chip removal.

A way to grip the chips, depending on how they are packed next to each other. Components, small vise grips, a small screw driver and a fine point awl should be all that are needed.

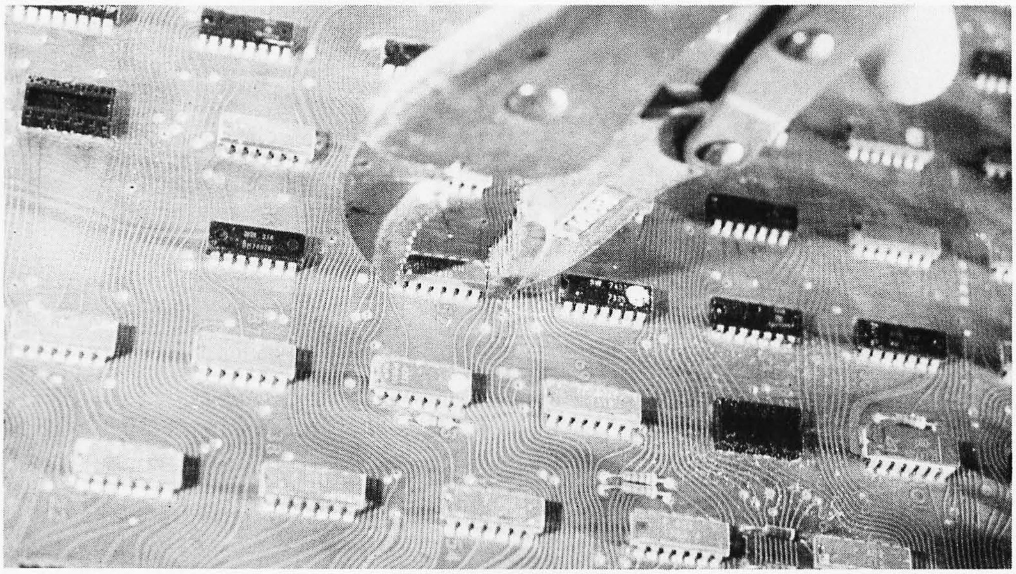
A place where splashed solder will not be serious.

Some form of eye protection.

WARNING 1

Using this method involves heating PC boards to high temperatures. Some boards release Hydrogen Chloride (HCl), which becomes hydrochloric acid in your lungs. Do this only in a well ventilated area, and stop to allow air to clear if irritation develops.

Grip the IC a second after removing the flame and rock it away from the board. It should come free in a couple of seconds.



WARNING 2

When an IC is pulled from a board, the board often snaps back to its original position. This is especially true if it isn't fixed very rigidly in place. When the board flips, solder is often sprayed away from the back side of the board. I ruined a pair of pants by not considering this before I started. I, therefore, wear old clothes and if you don't want solder on the floor, cover it with newspapers.

Enough warnings . . . following is how I pull ICs from boards:

First I clamp the board to my bench so that I can get my vise grips on about half the ICs (this is with a 10" x 14" board). I adjust the vise grips so I can grip a 14 pin IC without the vise grips locking and then light the torch. The flame on my Benzo-matic torch with the narrow tip is about an inch long.

Beginning with the lowest IC I can reach, I heat it with the torch by sweeping the torch over its pins (you obviously heat the *non-component* side). Especially when using a torch with a narrow flame it is necessary to move the flame over the pins. One complete sweep should be done once or twice a second. After a second or so, the IC should be gripped, and rocking tension away from the board applied. It helps to rock the IC, especially if corner pins have been bent over to hold the IC in place during assembly. The IC should very rapidly become loose, and in another couple of seconds should come free of the board.

When the IC is removed, quickly drop it on the bench and move the torch and pliers to the IC above the one removed. Heating the lower IC pre-warms the board above, making the next removal easier. Also, the board position just heated will cool faster, thereby reducing the amount by which the board will be damaged.

As each column of ICs is removed, the next is done. When all ICs on one half have been removed, reposition the board so the other half is accessible. I've found that the half-way point often can be a good excuse to let the room ventilate and drink a beer.

No matter how carefully and rapidly I've worked, I always burn the board at least once because I have trouble removing an IC, or my pliers slip, or for some other reason. If you consistently burn each board position, your flame is probably too hot. If, however, it takes longer than 5 to 10 seconds to remove an IC, your flame is too cool.

A certain amount of care is necessary when gripping the ICs. Too much pressure may crack them. Too little pressure will let the pliers slip, costing time to reposition them and marring the cases.

When attempting to remove the larger ICs such as 74181s and 74154s, which come in 24 pin DIPs, I have trouble gripping them, so I remove them as a two step process. First, I place an awl under the middle of one side, say between pins 6 and 7. I heat that pin row and, with the awl applying leverage, pull out that row. I then grip the IC on its thinnest dimension, heat the remaining pins, and remove the IC.

So far, by using this technique, my friends and I have removed about 1000 ICs from surplus boards which have about 80-100 ICs each. I tend to break 2% of the chips I pull by applying too much force with the pliers. But a friend has never broken one, so it clearly is an individual matter. Of those chips removed unbroken, we have tested around 250, and have never found a bad chip.

As an unrecommended demonstration of the ruggedness of ICs, I accidentally grossly overheated one, so that when I gripped it in vise grips, the chip was bent in a curve. The plastic case must have softened significantly. After allowing it to cool several minutes to the point where I could handle it by hand, I plugged it into a circuit, expecting it to have failed totally. It worked, although I didn't check out its ac characteristics. Out of general paranoiac distrust for a device so intensely mistreated, I discarded it.

After removing ICs from boards it is usually necessary to clean and straighten the pins. Boards with plated through holes often lose their plating around the IC lead.

I have found this method useful as a means of quickly building a stock of ICs ready to use in any project. It is limited mainly by the availability of exotic surplus chips, but most standard 7400 series TTL is easily available. The price of 4 cents/chip can't be beat, and the time required – about 10 to 20 minutes/80 chip board – is rather small.

This technique provides a fast, cheap, safe means of removing chips. I hope it proves as effective for you as it does for me.

SERIAL

We all know microcomputers use parallel data for internal operations. Computers - mini, micro or maxi - are often specified with the "bus data width" as a key parameter. This is the number of parallel bits which participate in operations at one time. Typical microprocessors now available have bus data widths of 4,8,12, or 16 bits. If you employ a used minicomputer in your system you may enjoy a 12,16,18, or 24 or even 32-bit wide bus. When shuffling data to or from memory and peripherals, the parallel lines of the bus are defined simultaneously - and you have to run at least as many physical wires to each interfaced subsystem.

When wire is at a premium you can get by with only one channel if the data is sent in a time-ordered sequence, one bit at a time. Communications and peripheral interfaces thrive on diets of serial bits provided the speed is relatively low.

In this article, Don Lancaster provides us with an excerpt from his forthcoming book, **TV Typewriter Cookbook**, to be published by Howard W. Sams, Indianapolis, Indiana. Don describes the basics of parallel to serial conversion and its inverse, using UART technology to do the transformation. His article this month concerns UARTs and serial interfaces which are relatively self-contained - local wires, tape recorders, etc. He also covers the communications aspects of serial data...radio and telephone network modem hardware.

While Don wrote the article from the point of view of the TV Typewriter technology which he pioneered, the problems he discusses are just as applicable to the home brew computer context...simply read "computer" every time you see TVT in the text. Don's comments on the serial cassette interface will provide one input to the discussion of various possible recording interfaces in the pages of BYTE.

... CARL

by
Don Lancaster
Box 1112
Parker AZ 85344

Most TV typewriter circuits need all their ASCII character and command bits simultaneously available in parallel form. This is also true of most electronic keyboard encoders and the bidirectional data buses of many minicomputers and microprocessors. In simple systems, we can connect all these parallel sources and loads together as needed without any further interface circuitry.

Sometimes, it's far more useful or convenient to have the bits march by one by one in *serial* form. While serial form is much slower, it has one big advantage — only a single wire or communications channel is needed, instead of multiple signal lines. Another benefit of serial form is that it can be made slow enough to communicate over ordinary phone lines, cassette tapes, radio channels, or electromechanical teletype systems. Some of the places where we'd like to use serial transmission are:

Remote Keyboards, where a single pair interconnection gets used instead of expensive multiple conductor cable.

Teletypes, where the bits have to be converted to signals based on current or no current in a wire loop.

Industry Standard Interfaces, such as the RS232-C and the newer RS422 and GPIB, allowing signals to travel relatively long distances.

Cassette Recorders, where we can store and exchange characters and programs with properly designed single channel, speed independent circuitry.

Radio Transmission, where only two tones on a single transmitted frequency are often used. This is typical of ham RTTY.

Modems, or Modulator-Demodulators that let us exchange data over the telephone line, either one way, or both ways at once.

We can call the circuits

INTERFACE

that get us from parallel to serial and back again *serial interface*. Usually, there are two distinct parts to the interface problem. The first is to convert from parallel to serial (or vice versa) at *logic level*, staying compatible with CMOS or TTL integrated circuits. This is often done in an industry standard single integrated circuit called a UART, short for Universal Asynchronous Receiver Transmitter. UARTs will simply and cheaply do the conversion and back again, along with providing all the necessary housekeeping bits, control signals, and noise immunity provisions.

The second portion of our conversion process gets us from logic levels to whatever form of signal the serial part of the system uses – such as dc currents for teletypes, bipolar signals for standard interface, and carefully selected tones suitable for cassette recording or transmission over a radio channel or a phone line. We'll be taking a detailed look at most of these techniques later.

How Fast?

There are two basic types of serial transmission we can use, synchronous and asynchronous.

In synchronous transmission, all the characters (called "words") are locked into system timing. We know the exact time position of each piece of data. If some time is to go by without doing anything useful, do-nothing words called *nulls* are provided. Timing signals must somehow be supplied to each end of a synchronous serial data system so we can tell when each word is to start. This usually means a separate timing channel or track or some sort of elaborate timing recovery circuit. Synchronous systems are usually fast and complex, but they are rarely used in most TV Typewriter applications.

With asynchronous transmission, the data words are not locked into system timing and can arrive with almost any spacing between words. To tell the beginning and end of a word, we have to add some new bit groupings, called *start* and *stop* bits, to the data. We don't have to provide any other locking signal between the source and destination of the data. Asynchronous data is commonly used in TVT systems.

Both ends of a serial transmission system have to *exactly* agree on a system speed, usually called the *Baud Rate*. The Baud rate is simply how many bits per second are going to be transmitted, including any start and stop bits. Fig. 1 shows us some common Baud rates. The most popular of these include 110, 300, 600 and 1200 bits per second. 1200 BPS is usually the fastest that can be handled by the phone company without fully dedicated lines. Inside systems, and where special lines can be provided, faster synchronous standard Baud rates of 2400, 4800 and 9600 bits per second may be used.

Should the Baud rate change between transmission and reception, such as when two recorders are used, or the batteries on one recorder age, the receiving end of the system has to be able to adjust itself to the new effective Baud rate if errors are to be avoided.

110 Baud is very popular for limited speed TV Typewriter uses. This rate is compatible with the ASR-33 eight bit Teletype code and corresponds to a 100 word per minute typing rate. While this is the fastest that most teletypewriter systems can be driven, and is easily handled by two-way 103 style phone modems, it takes painfully long to fill the screen. Even with a 512 character screen,

Serial Uses

- **Remote keyboards**
- **Teletypes**
- **Cassette recorders**
- **Modems**

Fig. 1. Standard serial communications speeds.

BAUD RATE	TYPE	TELETYPE COMPATIBLE?	DDD PHONE COMPATIBLE?	FRAME UPDATE COMPATIBLE?	TIME TO LOAD 512 CHARACTERS
110 BPS	Asynchronous	YES	TWO WAY	YES	51.2 SECONDS
300 BPS	Asynchronous	NO	TWO WAY	YES	18.7 SECONDS
600 BPS	Asynchronous	NO	ONE WAY	MAYBE	9.3 SECONDS
1200 BPS	Asynchronous	NO	ONE WAY	NO	4.6 SECONDS
2400 BPS	Synchronous	NO	NO	NO	2.3 SECONDS
4800 BPS	Synchronous	NO	NO	NO	1.2 SECONDS
9600 BPS	Synchronous	NO	NO	NO	0.6 SECONDS

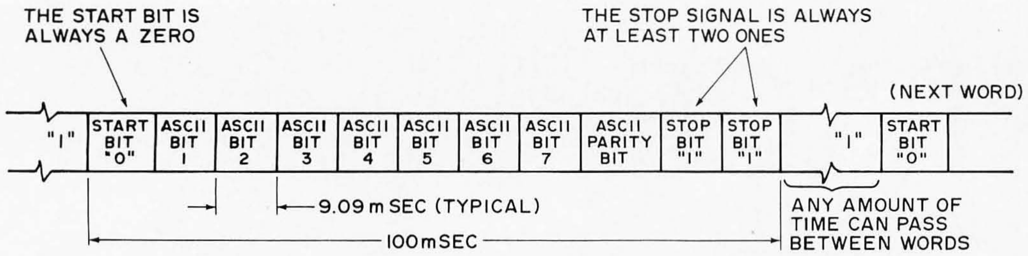


Fig. 2. 110 Baud, 100 word per minute code. "0" is a space or an open line. "1" is a mark or a shorted line.

UART = Universal Asynchronous Receiver Transmitter

RTTY = Radio Teletype

TVT = Television Typewriter

it takes 51.2 seconds, or almost a minute to load or retransmit the screen.

300 Baud is equal to 30 characters per second or 300 words per minute. This rate is the fastest normally used by a two frame update cursor system. It is also a rate easily handled by a cassette recorder, and by many two-way or full duplex modem systems. About 18.6 seconds are needed to load or dump a 512 character screen. These frame update and retransmission rates can be minimized by using a "virtual" update where one page is viewed while the other is updated, and by creative use of carriage return commands to return after the last character of each line, rather than going on out to the end of the line.

Faster Baud rates usually take more in the way of circuit design, and are usually limited to one-way modem transmission, premium recording techniques, and a Direct Memory Access type of update in the TV Typewriter.

A 110 Baud Standard

The 110 Baud, 100 word per minute code is an industry standard for slow data exchange. It is compatible with the Model 33 and Model 35 Teletype systems, and other teleprinters using an 8 level code. The code takes 100 milliseconds to send a character. The next character can follow immediately or can be sent any time later. Fig. 2 shows us the standard.

The original Teletype notation still carries over to this code. A Mark is a digital "1", a shorted line, or a completed connection. A Space is a digital "0", an open line or a broken connection. Between words, the teletype line or digital output is constantly putting out "1"s or marks, and is thus marking time. One of the reasons this was originally done was so that any break in communications is immediately known.

There are eleven bits to the code. Each bit is an identical 9.09 milliseconds long for a total code time of 100 milliseconds per word. Each word begins with a start bit. The start bit is always a zero and tells the receiving circuitry that a new character is to begin. The start bit is essential since some ASCII words will begin with one or more "1"s and there is no way to tell a marking time "1" from a "1" bit in an ASCII character code.

The ASCII bits follow in sequential order, starting with bit B1 or the least significant bit. After the seven character bits, an eighth bit is sent either as a "1" or providing a parity check bit for the rest of the word. At least two stop bits must follow the word. The stop bits are "1"s or marks, and any number of additional marking "1"s can follow between characters. The stop bits give the receiving circuitry a chance to shut itself down and await a new word.

The receiver can be electronic in the case of a UART or electromechanical in the case of a teletype. Between words, the receiver just waits. Since the data transmission is asynchronous, the receiver has no way of knowing ahead of time when a new word is to arrive, so it has to wait for a new start bit before it can do anything. The arrival of this bit activates the receiver, which then goes through a sequential procedure that sorts out the bits, puts them in parallel form, and outputs them.

With a UART, sequential time intervals of 9.09 milliseconds each are electronically generated and the center of each interval window is tested against the incoming code to see whether a "1" or a "0" is received. These are accumulated in a shift register, error tested, and output as a parallel word at the end of the interval. The stop bits are used to reset and shut off the circuitry.

With an electromechanical teletype, the break in line current caused by the stop bit releases a one-turn clutch on a mechanical scanning commutator that goes once around in 100 milliseconds. It sequentially routes the incoming code to a group of scanning solenoid magnets. These set up the code in parallel form, and at the end of the word, the scanner resets and the code is typed or output on paper tape.

Tolerances

It's extremely important that both the transmitter and receiver are clocking bits out at the same 9.09 millisecond rate, and that nothing happens in the channel to speed up or slow down the bits. There are many possible sources of error. If the bit

positions jitter around or are differentially delayed to by any tone keying, the filtering, or channel response, we get a bias error. Bias errors put individual bits ahead of or behind where they actually belong. One source of bias in a two tone modem or cassette system occurs when one tone is delayed more than the other in any filtering circuit. This is called the group delay distortion problem.

If the basic transmission and reception rates differ so that the bits can get ahead of or behind where they're supposed to be, we have a *skew error.* Note that bias errors apply to individual bits, while skew errors are progressive, making each sequential bit decision that much more difficult to detect without error. We get a skew error if there is an absolute timing difference between transmitter and receiver. Cassette recording systems introduce a potential skew error if the record and playback rates differ. This easily happens with cheaper units susceptible to speed variations with battery voltage, and almost is inevitable if the recording is done on one machine and playback on a second. Recording skew errors are correctable if the recording signals are designed to include speed information, and the receiver is capable of using this information to speed up or slow down as needed to eliminate this error source.

How accurate do we have to be? This is easy to calculate. Assume temporarily that there are zero jitter and zero bias errors in the channel, and that we are using an electronic receiver that very narrowly samples for valid data. The last data bit we are interested in is the parity bit. The center of the parity bit is 8½ bits removed from the beginning of the start bit, a delay of 77.26 milliseconds. The receiver delay is also supposed to be 77.26 milliseconds. If our sampling is narrow enough, we can be up to just under half a bit slow or fast and still be able to read the parity bit without error. This corresponds to a time error of 4.53 milliseconds either way, or slightly over 5%.

But, this figure leaves no room for bias and jitter errors and doesn't give the slower electromechanical circuits enough time width to reliably respond to the incoming data. As a practical rule, *the receiver and transmitter bit times must match to well within plus or minus one percent.* It is absolutely essential to hold things this close for low-error communications.

A 300 Baud asynchronous timing system is very similar to Fig. 2 and uses the same eleven bit code of equally spaced bits. The only difference is that the per-bit time is 3.33 milliseconds, corresponding to a 300 Hertz clock rate. Optionally, only a single stop bit may be used. This rate is cassette and TV Typewriter compatible and may be used for full duplex (two way) operation in most modem circuits, but it is too fast for teletype use.

Using UARTs

Parallel to serial conversion and back again can obviously be done with CMOS or TTL circuits. Basically, you parallel load a shift register and serially clock out data or serially clock in data to a shift register and then latch its parallel outputs when the data is valid. By the time you add all the error testing circuitry, housekeeping bits, synchronization, and so on, the circuits tend to get specialized and complex.

Instead of this route, you can use an industry standard MOS integrated circuit called a UART for virtually any serial to parallel conversion and return process. Several pin compatible UARTs appear in Fig. 3. These are general purpose, programmable devices that let you select the number of start and stop bits, the word length, type of parity, and so on to suit your particular system. Dedicated UART-like devices are also available for use with specific microprocessors. The Intel 8201 and the Motorola 6850 are typical of these.

The standard UART comes in a 40 pin package, and has supply voltages of +5 routed to pin #1, -12 to pin #2, and ground to pin #3. The later versions of these devices are N channel types that need no -12 supply, and that pin is left unconnected. The General Instruments AY-5-1014 is one of these.

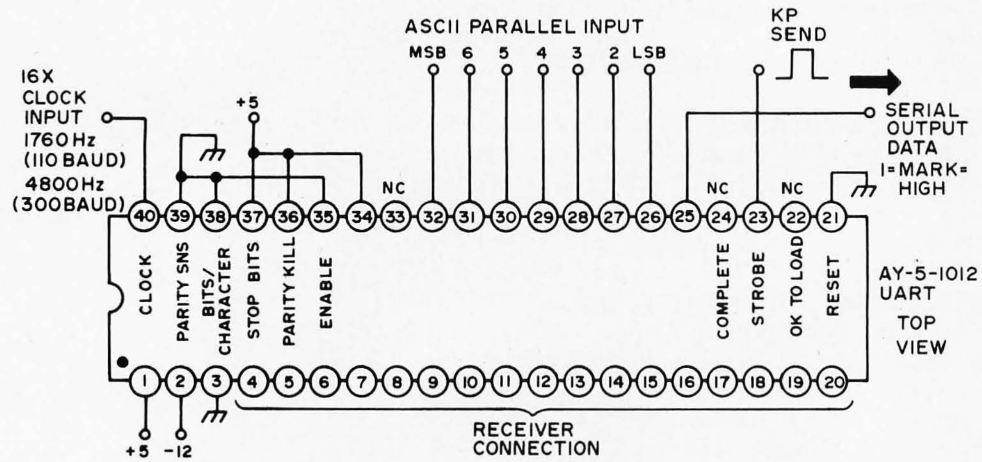
Fig. 3. Pin compatible UARTS.

S1883 (American Microsystems)
AY-5-1012 (General Instruments)
2536 (Signetics)
COM2502 (Standard Microsystems)
TMS6012 (Texas Instruments)
TR1602 (Western Digital)

The low number pins (1-20) are the receive portion of the UART, while the high number pins (21-40) are the transmit portion. Except for common word length and parity programming, the two halves of the circuit are separate, although they often are used as a send-receive pair.

Both the receiver and transmitter portions of the circuit need a clock. The clock frequency is usually *sixteen* times the Baud rate. This high frequency lets the UART do things like sample the center of each data interval and recheck for valid start signals and similar good things. For instance, a 110

Fig. 4. UART circuit to transmit code of Fig. 2.



Baud circuit needs a clock of 1760 Hertz, while a 300 Baud one uses a 4800 Hertz clock and so on.

The clock signals can be derived from a CMOS or 555 type of astable oscillator, but it is far better to digitally derive clock frequencies from TVT system timing or another stable source. Remember these clock signals must be held to well within one percent and ideally shouldn't have any adjustments. If our TVT has a 15,840 horizontal rate, we can divide this by nine to exactly get 1760 for a 110 Baud system. With a 15,720 rate, we get 1746.6 Hertz, a figure a bit low, but still useful and less than one percent under.

The receiver and transmitter clock inputs are on separate pins. They are often tied to a common clock source in simple send-receive circuits. One important exception is when a UART is used as part of a speed-independent cassette interface. In this case, the *receiver* clock frequency is derived from the tape during playback. While it is nominally the same as the transmitter frequency, its exact value is set by speed information recovered from the recorder. This can be used to eliminate much of the skew error that would normally result from a change in tape speed from time to time or machine to machine.

Fig. 4 shows us the connections for transmission of the 11 unit code of Fig. 2. The input ASCII code goes on pins 26 through 33, with the least significant or b1 bit on pin 26. A 16X clock goes into pin 40, and a KP send command goes to pin 23. The leading edge of this send command starts transmission, but the input data must be valid for the entire time the command is positive. Normally, this is a narrow pulse a few milliseconds wide, derived from a keypressed command on a keyboard. Serial output data appears on pin 25.

Pins 34-39 program the UART for different bit lengths and codes. 34 is an enable that normally remains high. Pin 35 provides a parity bit if it is grounded and omits one if it is high. Pin 36 picks the number of stop bits. Ground gives you one and high gives you two. 37 and 38 together decide how many *data* bits are to be sent, ranging from 5 to 8. Both grounded provide for 5 data bits, useful for Baudot RTTY transmission. The connection shown gives us a seven bit data word. Note that if you use the parity bit, it *adds to* the number of data words. The code of Fig. 2 uses our start bit, seven data bits, one parity bit, and one stop bit for an eleven unit code. Pin 39 picks even or odd parity with ground giving odd parity. An optional reset input is provided on pin 21. It is normally grounded. Bringing it high resets the UART. Without resetting, the first word transmitted after power is first applied can be wrong.

The UART transmitter is double buffered. This means you can load a new character as soon as the one already inside begins its transmission. Two optional outputs are provided. Pin 22 tells you when it is OK to provide a new character by going high. Pin 24 tells you that a character has been completely sent when it goes high.

There are two ways you can use a UART transmitter, either unconditionally or handshaking. In the unconditional mode, any time a character arrives, it gets sent. This is the simplest, but you have to make absolutely certain that characters don't arrive spaced or grouped too closely together. While pairs of inputs can be closely spaced much the same way that two key rollover works in a keyboard, you have to be absolutely certain that the long term average is never exceeded by the word rate of the UART. This means a 100 millisecond character spacing for 110 Baud, and around one third that for a 300 Baud system. In the handshaking mode, the UART decides when it wants to receive a new character, using the pin 22 and 24 outputs. Circuits driving the UART are set up to provide characters only when they are asked for them. For most TVT uses, unconditional UART transmission is simpler and easier to use.

Fig. 5 shows us the UART receiver circuit, again set up for the code of Fig. 2. The receiver logic has elaborate noise elimination provisions, made possible by the sixteen times higher clock frequency. Whenever a start bit is purportedly received, that bit is retested later and verified to prevent a random noise pulse from generating an unwanted character output. All data bits are narrowly sampled in the middle of their possible time slots, allowing considerable bias and skew distortion to exist without error.

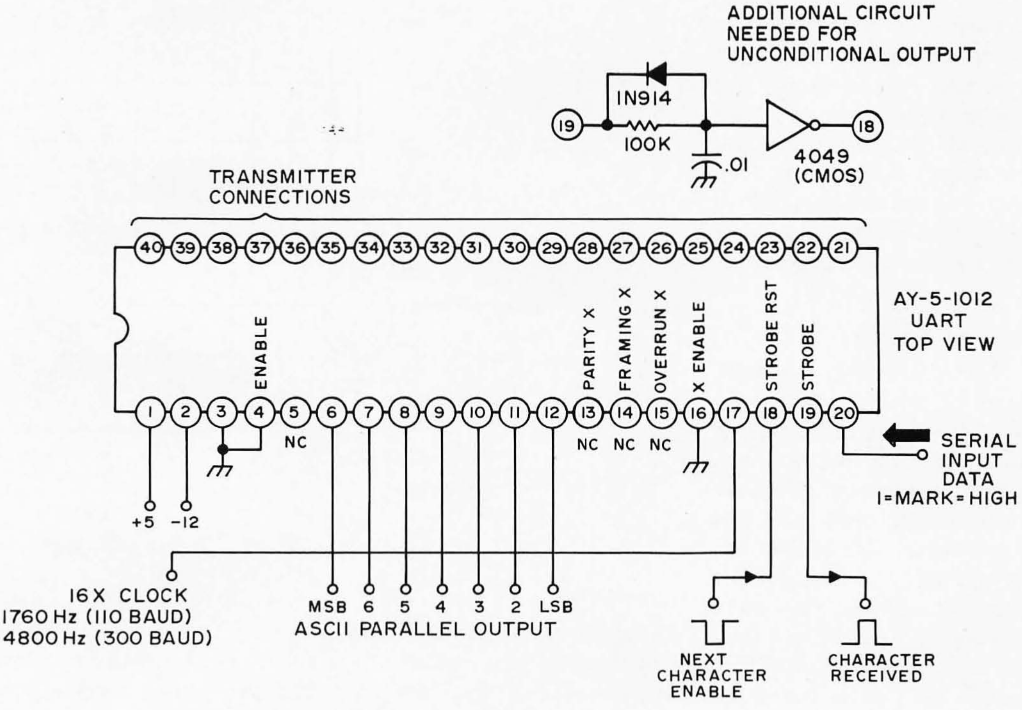
The same pin 34-39 inputs that programmed the transmitter's word length and format identically program Baud rate, applied to pin 17. In send-receive systems, we can often tie the receiver and transmitter clocks together. In speed-independent cassette interface circuits, the receive clock is reconstructed from speed information recovered from the recorder to eliminate skew errors.

The serial input is routed to pin 20 and converted to the equivalent seven bit parallel ASCII output on pins 6 through 12, with the least significant bit on pin 12. When the output is valid, a strobe on pin 19 goes high as an output.

This output strobe must be reset before a new character can be output. If we're operating in a invert it, and reapply it to the strobe reset input pin 18. Fig. 5 shows us one way to do this with a RC network and a CMOS inverter. Many UART receiver problems are caused by failing to reset the strobe after each character – watch this particular detail very closely.

Several additional outputs are available for use in fancier systems. Parity, framing, and overrun errors produce respective high outputs on pins 13 through 15. These can be used to ask for a repeat or put in a question mark to indicate a transmission error. All receiver UART outputs are tri-state and may be floated in systems where the UART's

Fig. 5. UART circuit to receive code of Fig. 2. Parity, word length, and stop bits set by transmitter programming, pins 34-39.



the receiver portion of the UART. Although the receiver and transmitter can be used in totally different circuits and at different baud rates, they have to operate with a common format, set by these pins.

The receiver needs its own clock of sixteen times its handshaking mode, the TVT circuitry accepting the character sends back an acknowledgement or completion signal that momentarily drives pin 18 low. If we are using an unconditional output mode, you somehow have to delay the pin 19 strobe output, outputs must share a common or a bidirectional data buss. Making pin 4 positive disables the ASCII outputs, while making pin 16 positive disables the error outputs.

Besides its usual use as a two way serial to parallel converter, there are other

useful circuit tricks you can do with a UART. For instance, by connecting the parallel outputs of the receiver back to the parallel inputs of the transmitter, you can change Baud rates. This is handy in speeding up slow data for use on a fast channel, and for correcting speed errors on cassette systems.

If the UART is to accept data from several sources you can either tri-state combine the sources onto a single input buss, or else use an input eight pole, double throw selector switch to pick one of two input channels. This is common in TVT service, where the keyboard forms one channel and the screen retransmission output buss forms the other. 4502 hex tri-state drivers or 4019 four pole selectors are suitable CMOS devices to use. In general, tri-state lines onto a common buss are much preferable to selectors, but few keyboard encoders have inherent tri-state output, and the output buss on the TVT often has to continously drive the display if we want to view the retransmission process.

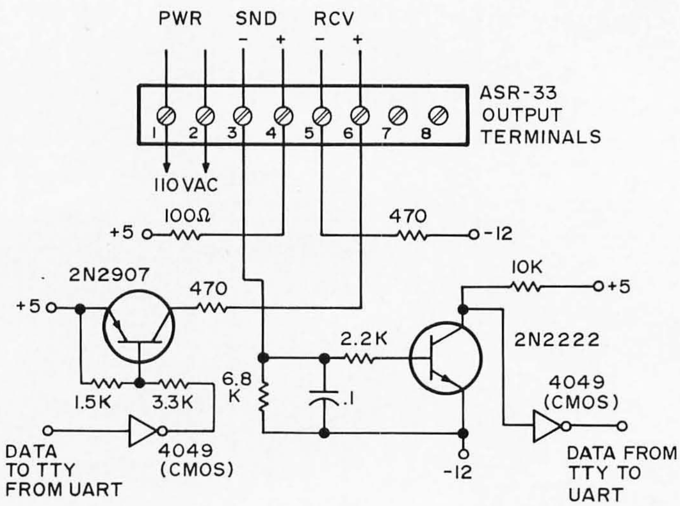
Teletype Interface

There are two common types of teletype systems in use today. The older type is the five code bit machine, typical examples of which are the Teletype model 28, the Creed model 75, and various Kleinschmidt models. While these machines are commercially obsolete, they still see usage for ham RTTY and some deaf communications systems. Their reasonable price availability makes them attractive for home computer hard copy as well, although the available character presentation is extremely limited. These older machines all use the more or less obsolete Baudot code and are not directly ASCII and TVT compatible, unless conversion ROMs and figures-letters logic is added to them. We'll note in passing that a UART may be used in the Baudot code by applying the code to pins 26-30 with the least significant bit on pin 26, and making pin 35 high and grounding pins 37 and 38. Older UARTs will generate two stop bits, while more recent ones (such as the TR1602B) will automatically generate the needed 1.42 stop bits in this mode.

The second common type of machine is the computer and timesharing standard teletype — the Teletype models 33 and 35, particularly the ASR-33. These use a standard eight bit ASCII code and follow the format of Fig. 2. They are directly compatible with TVT system coding.

Either type of system is based on breaking current in a dc loop. This current is often either 20 or 60 milliamperes. Transmission occurs when a mechanically coded commutator generates an output code by once-around breaking the current as often as needed.

Fig. 6. UART-teletype (ASR-33) interface. Teletype must be internally set to 20 mA loop current and full duplex operation.



Reception reverses the process. A momentary break representing the start bit releases a once-around commutator that distributes the code breaks to magnets which set up a pattern for printing when the scan is complete.

Fig. 6 shows us the interface for a Model 33 teletype. For optimum use, the teletype is internally programmed to a 20 milliampere current loop and full duplex operation. This is done following the teletype's maintenance manual. Full duplex operation means that the keyboard and printer aren't connected to each other. The keyboard can send and the printer receive both at the same time.

The transmitter interface provides a 20 milliampere current for a mark or a one and an open circuit for a space or a zero. The receiver senses a closed contact for a mark or a one and an open contact for a space or a zero. Extra inverters are added as shown to make the codes correspond so that a "1" from the UART is read as a "1" by the teletype. Be sure to observe the line polarities shown. The transistors can be almost any medium power, reasonable gain devices. More information on teletype interface appears in the Intel MCS-8 Users Manual.

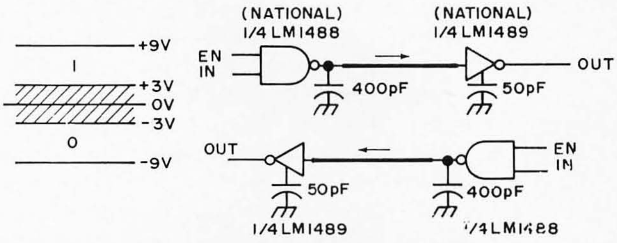
Similar current-no current interface loops can be used with older teletype systems. However, some of these machines need substantially higher currents and are notorious as transient and ground noise generators. With these older machines, total isolation is strongly recommended, using small, high speed reed relays or else opto-isolator circuits.

With any teletype system, the transmit Baud rate of the UART must match the needs of the teletype to within one percent. In the case of the ASR-33, a 110 Baud rate is needed, resulting in a 16X UART clock of 1760 Hertz.

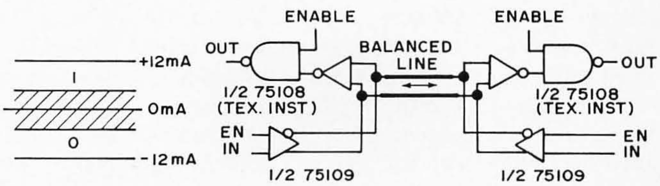
Industrial Interface

There are presently quite a few "standard" interfaces used to get between

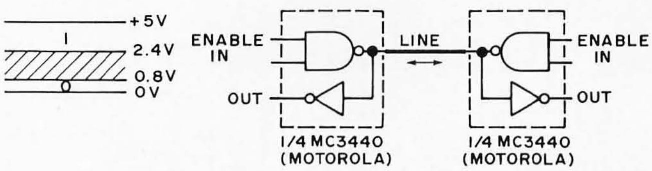
Fig. 7. Industrial interfaces for cable interconnections.



(a) RS232-C



(b) RS422



(c) GPIB

commercial modems, printers, test equipment, computers, and anywhere else you need a reasonable distance, noise free interface. Three of the most common include the RS232, the RS422, and the General Purpose Interface Buss or GPIB. Fig. 7 shows us what's involved in the way of signal levels and interface techniques.

RS232-C is an old EIA standard that predates IC techniques and is somewhat unwieldy. It is widely used in commercial modem circuits and for most large scale computer serial interface. The signal is bipolar, with a logic "1" being defined as +3 to +9 volts or more and a logic "0" being defined as -3 to -9 volts or more. Capacitors on the drivers limit risetimes to 30 volts per microsecond or less to minimize ringing and transient effects. Capacitors on the receivers limit the response as needed to reject noise but pass the highest transmitted Baud rate. The 1488 and 1489 integrated circuits form a typical interface pair.

RS422 is a newer EIA standard that uses balanced transmission lines and differential current sensing to eliminate any common mode noise. A current, typically of 6 to 12 milliamperes in one direction, defines a "1", while the reversed current defines a "0". The balanced line may be used either single direction or bidirectional, depending on how the receiver enables are used. In any bidirectional or party line system, it's extremely important to enable only one driver at a time. The 75108 and 75109 are typical balanced line drivers and receivers.

The General Purpose Interface Buss or GPIB uses TTL compatible levels, but combines them with a terminated line, high drive capability, and receivers with hysteresis for good noise immunity. The GPIB is most often used in parallel form to interface test and measuring equipment with computers and calculators. It can also be used as an effective serial interface that takes no special power supplies. The 3440 is a quad bus transceiver useful for GPIB service.

More information on EIA standards are available from the Electronic Industries Association, 2001 Eye St. NW, Washington DC 20006, while information on the GPIB interface is available from Hewlett Packard, 1501 Page Mill Rd., Palo Alto CA 94304.

Cassette Interface

Magnetic storage in the form of tape drive, disc files, and floppy discs have long been a standard and expensive way of storing bulk serial data for computer use. An obvious and extremely low cost substitute for these would seem to be the ordinary audio cassette recorder. Besides providing bulk storage, the cassette can replace paper tape and punched cards, and handle programs as well. One big advantage of cassettes is potentially low cost duplication and distribution and exchange of programs.

Cassette recorders present several serious design problems when used for storage of digital data. Even a quality machine will vary its speed by a percent or more, and the machine to machine variations, particularly on lower cost units, can far exceed the amount of skew

Fig. 8 Speed independent cassette standards.

110 BAUD (CODE PER FIG. 2)
ARK = 1 = 16 CYCLES OF 1760 HZ
SPACE = 0 = 8 CYCLES OF 880 HZ

300 BAUD (FIG. 2 CODE WITH 300 HZ BIT RATE)
MARK = 1 = 16 CYCLES OF 4800 HZ
SPACE = 0 = 8 CYCLES OF 2400 HZ

600 BAUD (FIG. 2 CODE WITH 600 HZ BIT RATE)
MARK = 1 = 8 CYCLES OF 4800 HZ
SPACE = 0 = 4 CYCLES OF 2400 HZ
(clock doubling required)

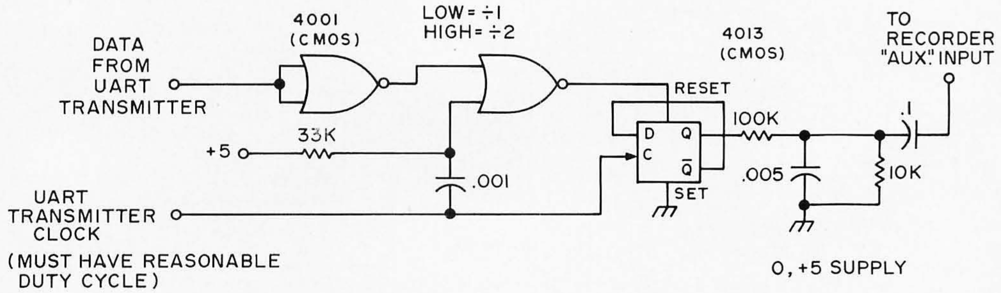
distortion the code of Fig. 2 can stand – if the receiver UART is running at a constant clock rate. Some of the techniques used to send data over modems and radio channels will work with a single, quality recorder, but these circuits are far from optimum as they have no way to compensate for recorder speed variations. Similarly, many of the standard computer tape recording techniques may work, but they are based on a different type of recording head and system – one that works with pulses, saturated flux changes, and sense amplifers, rather than one that records and plays back sine waves more or less linearly.

An ideal cassette interface would handle any machine and provide for machine to machine variations. We'd also like to supply some protection against dropouts, perhaps in the form of integration or multiple voting on what constitutes a one or a zero. A low cost, simple system using a single supply voltage and a minimum number of non-critical adjustments is obviously desirable, particularly if the system can be very tolerant of recorder levels and settings, and need no exotic codings, preambles or other limitations.

We can base such a speed variation tolerant or "speed independent" system on a unique property of the UART. For every received one or zero bit, the UART needs exactly sixteen receiver clock pulses. By designing a standard so that sixteen cycles of clock are recovered from a one recorded on the tape, and sixteen cycles of clock are recovered from a zero on the tape, the UART will always receive just the exact number of clock pulses it needs per one or zero. Tape speed variations of plus and minus 30 percent or more should be possible.

Fig. 9. Speed independent cassette interface. Values shown for 300 Baud rate. Recorder speed may vary ±30%.

(a) Record circuit.



(b) Playback circuit.

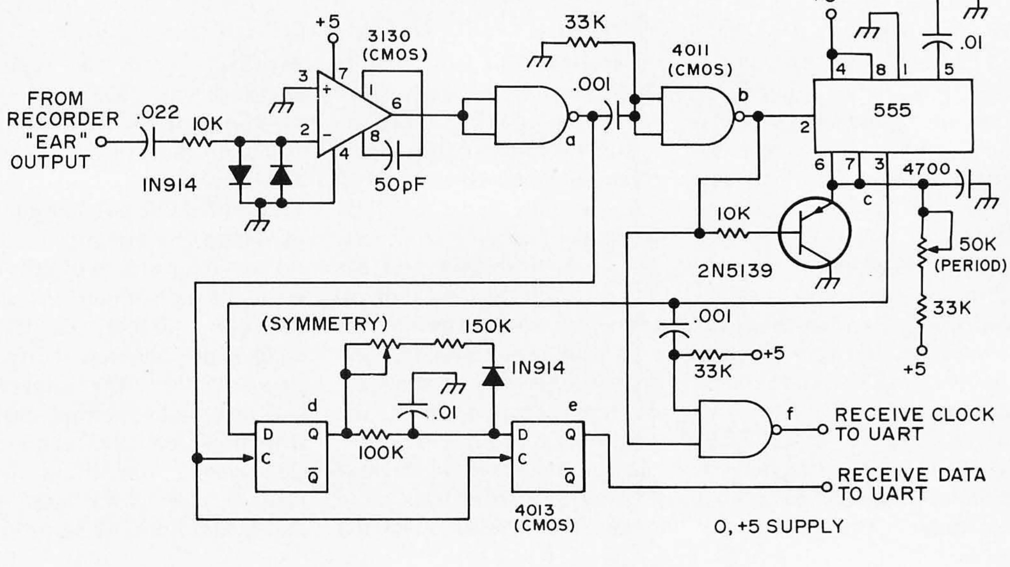


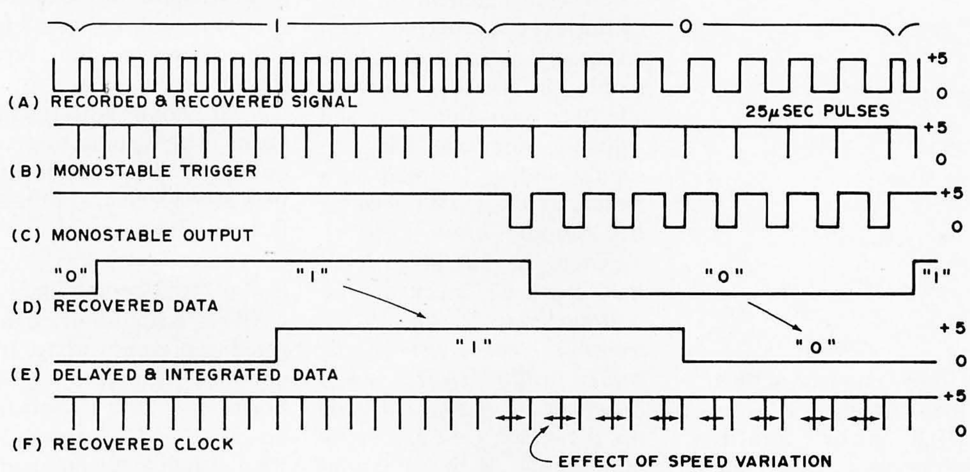
Fig. 8 shows us a set of standards for speed independent cassette interface. All three standards are based on the code and bit timing of Fig. 2. A 110 Baud standard records 16 cycles of 1760 Hertz for a one and eight cycles of 880 Hertz for a zero. The 300 Baud version, which is recommended for most uses, records 16 cycles of 4800 Hertz for a "1" and 8 cycles of 2400 Hertz for a zero. A 600 Baud version can be based on the 300 Baud standard by halving the length and doubling the clock recovery.

Fig. 9 shows us the 300 Baud circuitry involved, along with the key waveforms of Fig. 10. Even including the $5 to $10 cost of the UART (usable elsewhere in the TVT anyway), the circuit is extremely simple, non-critical, and cheap.

Fig. 9A shows the record circuit. As usual, a UART transmit clock of 16 times the Baud rate must be provided. For recorder use, the duty cycle of this clock must be stable and nearly 50%. Note that the UART serial data will always change synchronously with the UART clock, after a brief propagation delay. Clock and Data from the UART transmitter are sent to a gate and flip flop that divides the clock by two if the data is a zero and divides by one if the data is a one. This is done by resetting the flip flop about half way through a clock cycle if the data is a "1". We get sixteen clock frequency cycles with a "1" and eight half clock frequency cycles with a "0". Since the inputs are synchronized, there are no transient problems, and all cycles are full length. This output is 10:1 attenuated and moderately filtered to round the rise and fall times. This prevents excess peaking of the high frequency compensation inside the recorder. The output is capacitively coupled to the recorder AUX input.

The receiver gets its signal from the recorder EAR output. This signal is high pass filtered and doubly limited, first by a pair of diodes and then by a CMOS op amp. This particular op amp lets you run the inputs at the same voltage as the "negative" supply. If you use a standard op amp here, you'll need either a negative supply or positive input bias.

Fig. 10. Key waveforms of speed independent cassette interface.



A CMOS gate following the op amp improves the rise and fall time.

The gate's output is shortened and generates a pulse that lasts for 25 microseconds or so, coincident with the positive edge of the transmitter clock. This trips a negative recovery monostable set to two thirds the period of the lower frequency clock. If a string of "1"s is received, the monostable keeps on being retriggered and never drops its output. If a string of "0"s is received, the monostable drops its output for the final one third of each cycle. The net result is that you get a train of 8 negative going pulses out for each zero and nothing for each one. The leading edges of these outputs are sensed and shortened to 25 microseconds, and then combined with the input 25 microsecond pulses. The result is 16 clock pulses routed to the UART receive clock – either sixteen from the data for a one, or eight from the data and eight from the monostable for a zero.

The monostable's output also goes to a flip flop to recover the data. We get a "1" out of the first flip flop whenever 16 cycles of high frequency data are received and a zero whenever 8 cycles of low frequency data are received. The second flip flop integrates the output of the first one to eliminate any noise pulses and to take an average of several sequential ones or zeros before providing an output.

As the tape speeds up or slows down, the input square waves will also speed up or slow down. The monostable's output can absorb almost a 33 percent increase (from half to 2/3) of the high frequency clock period, or a 33 percent decrease (from 1 to 2/3) of the low frequency clock period without error.

The spacing of alternate zero clock pulses to the UART will change as the speed changes — but the UART doesn't care about this so long as the clock pulses don't actually overlap. This jitter is automatically eliminated by circuits inside the UART.

You calibrate the system by inputting a string of "1"s and noting the pot position where errors first happen. You then input a string of zeros and note the pot position where errors first happen. Then you set the pot one third of the distance from the limiting one to the limiting zero settings. This is a non-critical adjustment. The symmetry control can optionally be adjusted with a scope, or else by inputting alternate ones and zeros and adjusting for one half the supply voltage (as read on a meter) at the UART receiver data output.

There is one little detail that must be checked. *The positive edge of the UART transmit clock must be the same edge that, when received, is tripping the negative recovery monostable.* If it isn't, you may get partial pulses instead of a nice uniform eight pulse train during a zero. Recorders may vary in their internal circuitry and may provide a phase inversion between input and output. If you have this problem, either use the $\overline{Q}$ output of the recorder flip flop, or interchange the connections on pins 2 and 3 of the receiver limiter. Once set for a given recorder, there should be no further problems along this line.

Oddly enough, though the circuit will work with poor quality recorders, the tape quality should be good — bad tape means bad data.

While the circuit seems to work with amazingly poor recorders, best operation is gotten with a clean, medium to better grade recorder, preferably one that has an automatic level control on the input and tone controls available, along with an AUX input and an EAR output. Best operation will normally result with the output volume control about half way up and the tone control set to maximum treble boost.

A very important point that's often overlooked is the tape quality. Bad tape means bad data. Use only fine grain premium tapes (Radio Shack Supertape is typical). The key thing to watch for is whether the amplitude variation is guaranteed to less than one decibel. If the variation isn't specified — *don't use the tape.* The cost difference between good and cheap tape is negligible. You should also always "certify" your tape before you use it by writing a repetitive and obvious pattern over the entire tape length and checking for errors, splices or dropouts.

Extra controls to start and stop the recorder can optionally be added to make the storage more efficient. Data should never be entered or removed until the recorder is up to speed, and the UART's output lines can be gated to prevent garbage from getting into the system. Formatting data by setting it up on the TVT screen and then using a screen read to load the tape will give you dense storage.

One final detail that may need watching is to make sure the cassette recorder can't *overspeed* the system if you happen to record on a slow machine and playback on a fast one. This could output characters faster than can be accepted by a teletype or a TVT frame update system. The way around this problem is to set some maximum possible character rate that will let the return characters speed up without problems — use of a 7.5 Hertz rate on a 110 Baud system or a 22.5 Hertz rate on a 300 Baud system is one example. A second potential solution is to use two separate UARTs — programming the transmitter for two stop bits and the receiver for only one. When going from a cassette to a teletype, data can be resynchronized by connecting the UART parallel receiver outputs to the UART parallel transmitter inputs. Once again, make sure you can't overspeed the system.

Radio Data Links

One of the more common methods of sending serial digital data over a radio channel is to use a two frequency *frequency shift keyer* method in which one frequency represents a digital one and the other a digital zero.

Ham RTTY provides us with a typical example. At the audio baseband, two tones are used, defined as 2125 Hertz for a mark, or one, and 2925 Hertz for a zero or space. These tones represent the fifth and seventh harmonic of 425 Hertz.

These tones may be digitally generated the same way the modem tones of the next section are produced, or may be generated by a voltage controlled oscillator such as a 555, 8038 or a 566. These frequency shifted tones are used to frequency modulate an rf carrier. Alternately, the carrier itself can remain at its normal frequency for a mark and can be shifted down 850 Hertz (the difference between 2125 and 2925) for a zero, with the audio differences being picked up by mistuning the receiver by 2125 Hertz.

Fig. 11 shows us a typical receiver demodulator circuit. The carrier is received and detected by a FM receiver, adjusted to output audio tones of 2125 and 2925 Hertz. These tones are limited and routed to two bandpass filters, one set to the upper and one set to the lower frequency. Outputs are amplitude detected and compared, resulting in a one out for a frequency of 2125 and a zero for 2925. This

output may be routed to a UART for serial to parallel conversion. Normally a 7.42 unit code of 60 or 100 words per minute, using Baudot encoding, is used for ham RTTY.

Any radio carrier system must follow the rules and regulations for the particular frequencies used. Best performance of a frequency shifted keyed system normally results when the generated frequencies are sinewaves and are switched, transient free, at their zero crossings. Receiver filters should delay both sets of frequencies identically to prevent ones from getting ahead of zeros or vice versa, and thus creating times when neither a one or a zero, or both of them together are simultaneously present. As with any serial interface, input and output code formats and Baud rates must closely agree.

Modems

Modems, or modulator-demodulators, are ways to get tones or tone groups onto the telephone line and off again in order to send and receive digital data. Two common ways of coupling modems to the phone line are to use small speakers to acoustically couple to a standard handset, or to directly connect to the phone line through a suitable protective network or data access arrangement. Acoustical coupling can be used anywhere on an unmodified telephone, but has problems with frequency response, microphonics, and second harmonic distortion caused by the carbon transmitter. Direct coupling gives better control and better performance, but has to meet certain telephone company regulations and interconnect restrictions, needs a physical connection to the phone line, and needs its own hybrid or means of separating transmitted and received data.

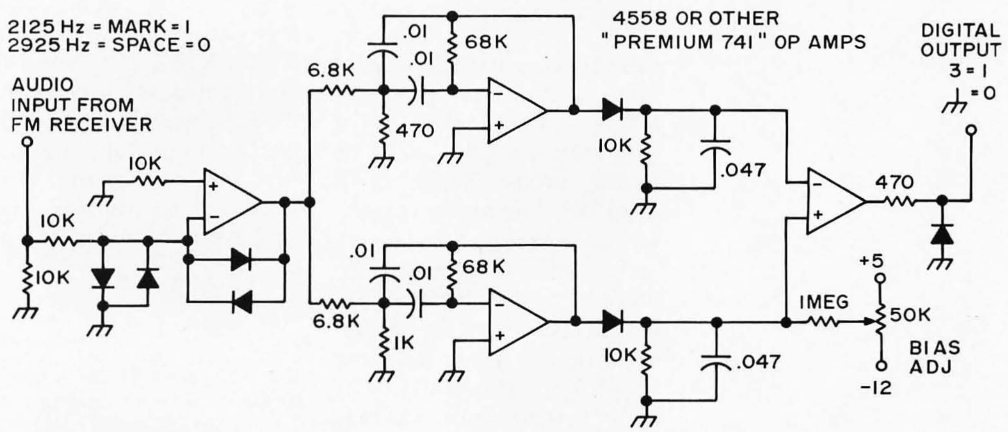
There are several basic ways to use modems. Simplex transmission goes one way only. Simplex with a back channel goes one way only, but provides for some low frequency communications, often limited to 4 Baud or less, in the other direction. This can provide for handshaking, message acknowledgement, etc..., but is far too slow to return data. Half duplex systems can send or receive data, but not simultaneously. Either the transmitter is off or in a mark condition while data is being received, or the receiver is disabled while data is being sent. In full duplex systems, data can be sent both ways, independently, and at the same time.

There are at least three basic types of modem systems that are useful over ordinary telephone lines, based on the Bell 103, 202 and 400 series systems. You can rent these from the phone company or others, buy them outright from modem firms, or design your own with the guidelines of this chapter. Most of the commercial modems use RS232-C interface standards and include such logic and switching functions as automatic answer and hangup, carrier detect, and other housekeeping signals.

The 400 systems are based on the touch tone ringing frequencies and accept contacts as inputs and are limited in the number of characters and the Baud rate. Baud rates of 10 to 20 characters per second are usually the maximum, and operation is normally simplex, with a separate unit needed for transmission and reception.

The 202 systems are half-duplex modems that can run up to 1200 Baud over the

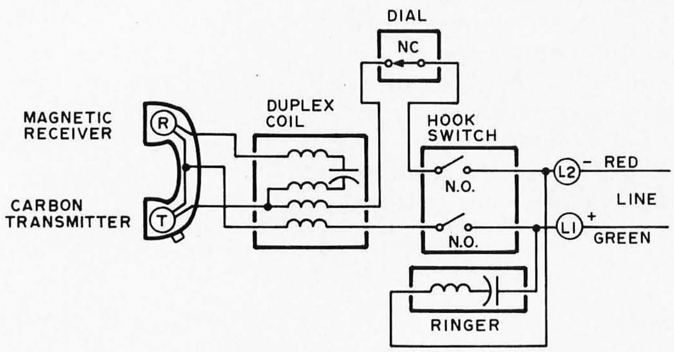
Fig. 11. Audio processor for RTTY receiver.



phone line, but cannot simultaneously communicate in both directions unless a special four wire system is used.

The 103 modems are full duplex and may be used at 110 and 300 Baud rates. For the majority of TVT uses,

Fig. 12. Telephone schematic.



this series, run at either Baud rate, is the most practical. Unlike the other serial interface circuits of this article, proper design of a good 103 style modem circuit with reasonable noise performance is a major job, particularly if your circuit has to operate over the dialup network for long distances and is to reliably communicate with commercial modems on the other end.

Phone Characteristics

A simplified schematic of a standard 500 telephone set is shown in Fig. 12. A carbon variable resistance transmitter and a magnetic headphone-style receiver are connected to the line by way of a duplex coil, a normally closed dial contact, and a pair of open when unused hookswitch contacts.

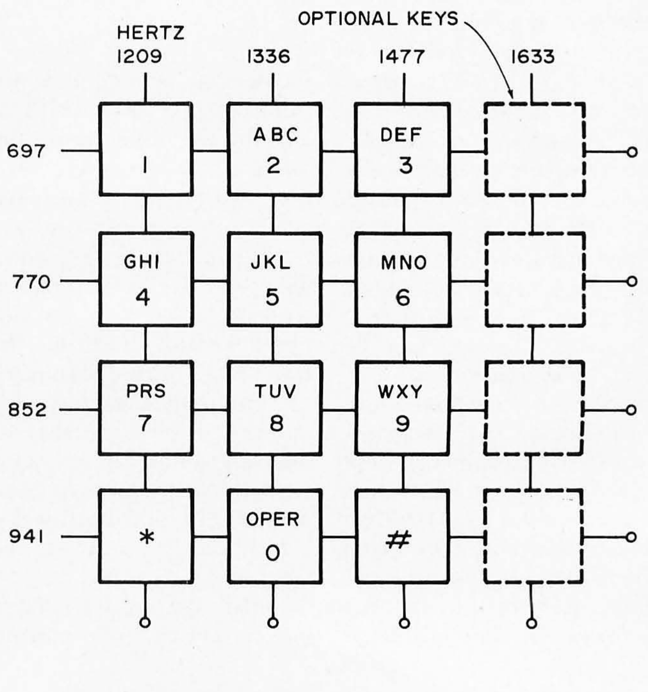
The duplex coil makes sure that outgoing signals reach the line and that ingoing signals reach the receiver with a minimum of interference. This is done by having two transmitter windings induce nearly equal and opposite signals into the receiver windings. This effectively cancels much of the local transmitter's signal into the receiver. The net result is to keep transmitter energy from being wasted in its own receiver and minimizes a "hear yourself" sidetone that psychologically makes people speak much more quietly. The duplex coil attenuates the transmitted signal by four decibels (to 60% voltage), the received signal by two decibels (to 80% voltage), and the sidetone by seventeen decibels (to 14% voltage).

The line is powered by a 48 volt central office battery supply, and the ac impedance of the line is nominally 600 Ohms, but varies with distance and quality of service. The audio signal levels at the line terminals are fractions of a volt. Normally, the loudest permissible modem tones are around a quarter of a volt, measured on the outgoing line. Received signals are lower still, typically one half to one tenth this value for local service, and even less on long distance loops.

When the phone is connected to the line by lifting the hookswitch, the line voltage drops to around 6 volts or so. The traditional dial signals by breaking this connection to deliver a group of mechanically spaced pulses that jump the line voltage between the open circuit and phone-off-the-hook values. Touch tone systems replace the dial with a low impedance that sums the tones of the next section (see Fig. 13) onto the line for signalling, a pair of tones at a time.

The ringer is capacitively coupled across the line and is resonant to some low frequency in the 20 to 47 Hertz range. Ringing voltage is an ac signal of 86 volts RMS. Selective ringing of a party line can be done in three wire systems with a ground return by ringing one phone from L1 to ground and the other one from L2 to ground. In two wire systems, ringer circuits with different

Fig. 13. Standard touch tone frequencies. Each key simultaneously generates two tones as shown.



resonant frequencies can be selectively rung by changing the frequency of the ring signal.

For direct entry modems, either a protective network or a data access arrangement such as the Bell CBS or CBT units can be used. These networks simulate the impedance of the telephone when activated and prevent any supply voltages from going onto or coming off of the power line. Under no circumstances should dc power be applied to or removed from the phone system lines, or any impedance be placed across the line or to ground that would degrade normal telephone services.

400 Style (Touch Tone) Modems

Modems based on touch tone signalling frequencies are usually limited to low data rates and often to a limited number of available characters. Touch tone signalling is based on simultaneously sending a pair of carefully chosen tones, following the code of Fig. 13.

The tone pair must exist for 40 milliseconds and the minimum time between tone pairs is 40 milliseconds, with a resulting maximum character rate of 12 per second. Touch tones are normally entered at signal levels somewhat higher than other voice and modem signals, being around three quarters of a volt RMS for the high frequencies and around half a volt for the low group. Line characteristics equalize these amplitudes by the time they get to recognition circuits.

A touch tone modem transmitter can simply be the touch tone dial of a remote phone, or it can be a circuit to generate two sine waves of proper amplitude and frequency simultaneously. Unlike other modems, and much of the serial interface of this chapter, the code is activated directly by contact closures. One closure, rather than a serial code, is all that is needed to send one of twelve or one of sixteen separate pieces of information. Additional tones or tone combinations can be added, such as in the Bell 401L or 402C systems that offer 99 or 256 characters.

Touch tone reception consists of three parts. First, the signals need sharply filtered with bandpass group filters whose response is 650 to 1000 Hertz for the low band and 1150 to 1700 Hertz for the high band. Adequate prefiltering is absolutely essential for most tone detection schemes. Tones are then detected, using limiters and slicers somewhat similar to Fig. 9, using narrow bandpass filters and detectors, or using phase lock loop tone detectors such as Signetics 567 tone decoder. Finally, the detected tones are combined with suitable two of eight digital logic.

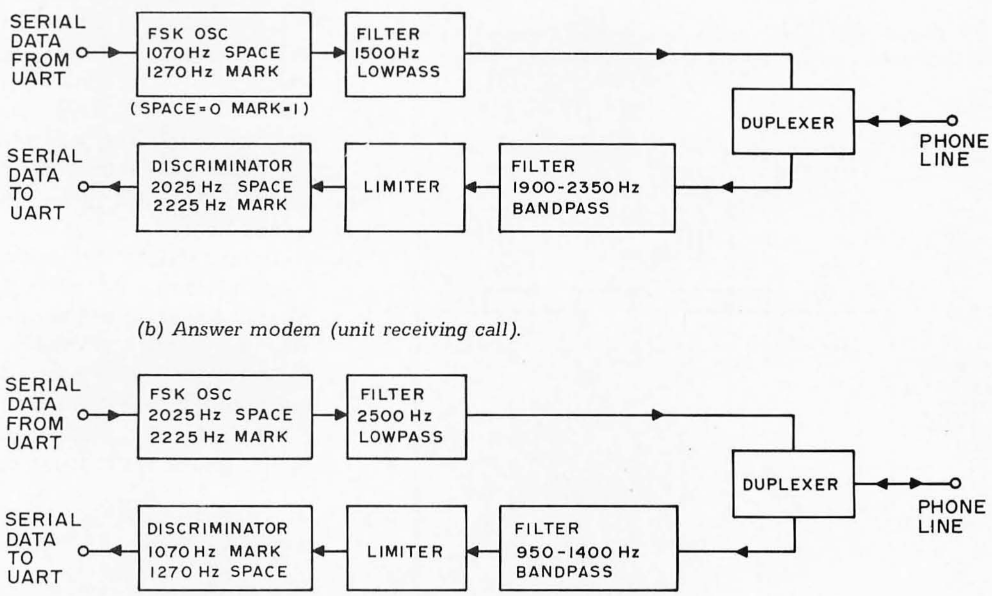
Additional information on touch tone techniques appears in the March 1963 *IEEE Transactions on Applications and Industry*, the Signetics *Linear IC Applications Manual*, and various issues of the *Bell System Technical Journal.*

103 Style (300 Baud, Full Duplex) Modems

"103" style modems are often the best choice for TVT use, as they offer full duplex, two way, operation at 110 or 300 Baud rates over the ordinary phone line. Fig. 14 shows us a block diagram of this type of modem.

The circuits are used in pairs. The modem at the end that's doing the calling is called an originate modem. It sends a 1070 Hertz sine wave for a space or zero and a 1270 Hertz sine wave for a mark or one, usually at a phone level of -10 DBM or around a quarter of a volt RMS. The modem that's doing the receiving is called an answer modem, and it

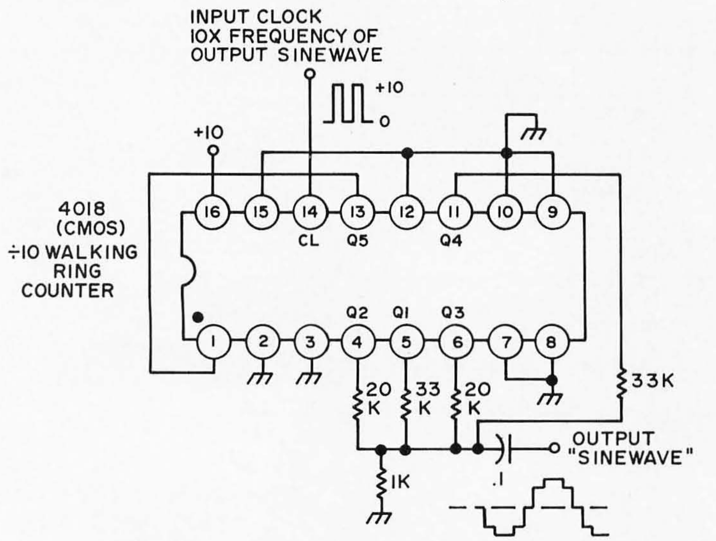
Fig. 14. Full duplex 300 Baud modems.



(a) Originate modem (unit making call).

(b) Answer modem (unit receiving call).

Fig. 15. Digital "sine wave" modem transmitter is easily filtered as first strong harmonics (-20 dB) are the ninth and eleventh.



receives and responds to these two frequencies. In turn, the answer modem transmits a 2025 Hertz sine wave for a space or a zero and a 2225 Hertz sine wave for a mark or a one. These, in turn, are acceptable to the originate modem.

These frequencies are carefully chosen to allow two-way conversation without interaction. The answer modem always transmits on the high frequency since a 2025 Hertz note is needed to automatically disable echo suppressors used on long distance phone lines, and to provide a standard recognition signal for automatic dialing equipment. (Echo suppressors effectively convert long distance lines into voice keyed one way lines. Two way transmission on a long line is not possible unless these suppressors are defeated.)

There are several very important things to consider when you are designing a modem. The transmitted signal must be a low distortion sine wave. Particularly, its second harmonic must be extremely low to prevent the originate modem's transmitter splattering its own receive spectrum with its second harmonic. When acoustical coupling is used, the transmit level must be held low enough that the rather bad second harmonic distortion of the carbon mike doesn't raise its signals to an intolerable level. Partial compensation of this carbon mike effect can be gotten by summing a sine wave of plus one half the third harmonic with the fundamental. This causes some cancellation and allows a higher level of transmission. Since most modem detectors use only the zero crossing information, it's important to coherently switch between these two frequencies, changing only when the sine wave goes through zero. The coherent operation eliminates "short cycles" that will jitter the received data.

Input signals to either modem *must* be strongly filtered to get rid of the other channel tones, as well as interference from speech, noise, touch tone coding, and other signals. The duplex coil in the phone set reduces but does not eliminate sidetone coupling. If you build your own duplexer instead, the same cancellation is only partial because of changing telephone line impedances.

In addition to getting rid of unwanted signals, there's a second severe restriction to the input filter. Both the ones and zeros going through the filter must be delayed an equal amount. Otherwise the ones and zeros will get out of step with each other and cause timing errors.

There are two basic ways to go about building this style of modem. An analog modem generates and decodes its signals using gated oscillators, RC networks, and phase lock loop detectors. A digital modem uses all digital logic for the frequency generation and detection. Analog modems should be avoided for several reasons. The transmitters inherently have less stability, need field

Fig. 16. Digitally derived modem frequencies.

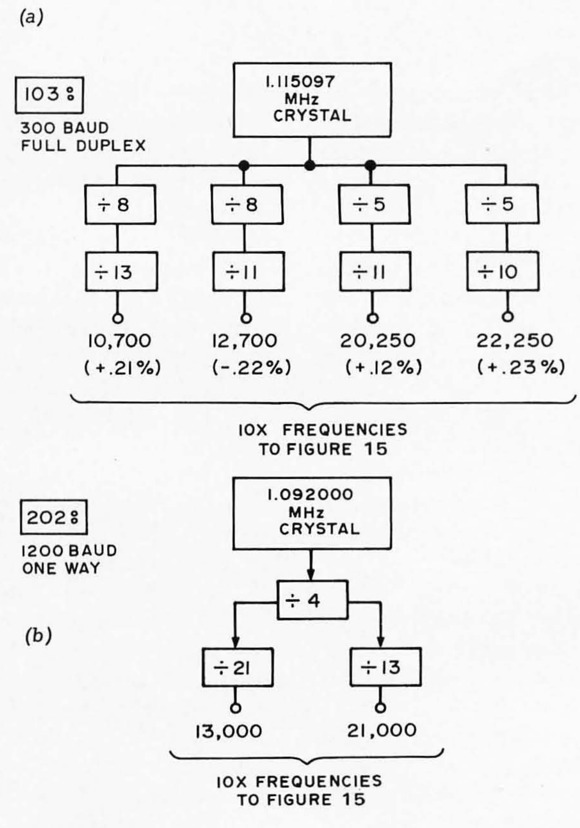
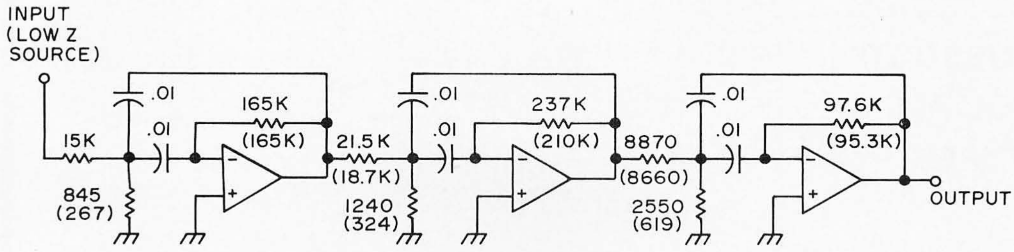


Fig. 17. Modem receive filter. 4558 or 7415 op amps. 950-1400 Hz = normal values = answer filter. 1900-2350 Hz = (parenthetical values) originate filter.



adjustments, and potentially have a stronger second harmonic, besides needing calibration. They are harder to coherently switch at zero crossings to eliminate transients. Analog receivers also must be calibrated and able to accurately resolve a small frequency difference, again with adjustment and calibration.

At this writing, there is no such thing as a modem on a chip. The Motorola MC6860 is one IC that handles approximately one fourth of the circuitry needed for a digital 103 modem. Two Exar chips, the 2207 FSK generator and the 210 FSK demodulator provide around half the circuitry for an analog system. A premium set of four hybrid integrated circuits from Cermetek Electronics is available that does the whole job in their Minimodem CH1213, 1214, 1252, and 1257 devices.

Figs. 15-18 show several techniques that might be of use in your own modem designs. Fig. 15 is a CMOS digital IC sine wave generator; it produces a sine wave in response to a 10X digital clock input. It is based on summing phases of a walking ring or Johnson counter and has negligible harmonic output up to the ninth and eleventh, which are both twenty decibels down (1/10 the amplitude) and easily filtered. The output can also be used to coherently synchronize input switching. Fig. 16 shows a digital timing sequence that starts with a crystal and produces all four modem frequencies needed for the Fig. 15 circuit. It can be built with a CMOS 4520 and a gate or two. Fig. 17 shows us some active filters useful as pre-filters with controlled group delay distortion. Fig. 18 shows an adjustment and calibration free receiver digital discriminator.

More information on 103 style modem designs are available in Motorola applications note AN731, Exar data sheets XR210 and XR2207, *The Active Filter Cookbook* (Sams), and Cermetek Microelectronics Minimodem data sheets.

202 Style (1200 Baud, Half Duplex) Modems

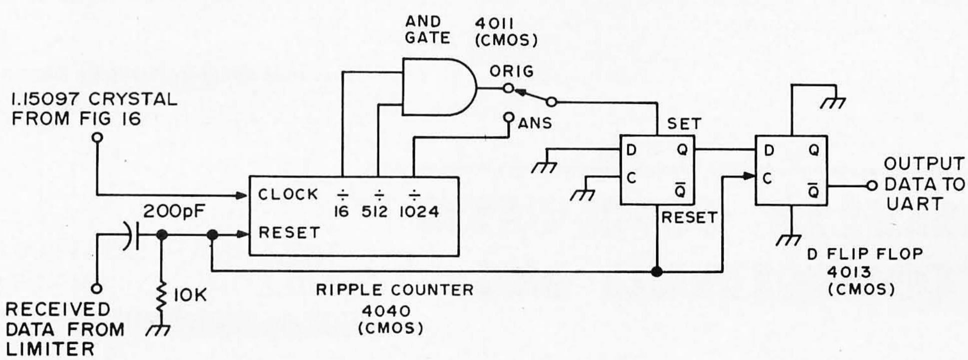
The 202 style modem circuits are both faster and simpler than the 103 versions and require less in the way of circuitry. Their big disadvantage is that most of them are strictly simplex or half duplex devices when used on the ordinary two wire phone line. Simultaneously transmitting and receiving is ordinarily not possible.

Like the 103, 202 standards use frequency shift keying. Bell standards call for a 1200 Hertz mark or one and a 2200 Hertz space or zero, while international standards call for a 1300 Hertz space or zero and a 2100 Hertz mark or one. An additional tone may need generation to provide for automatic answering. Commercial units also sometimes provide a back channel of four or five Baud for acknowledgement.

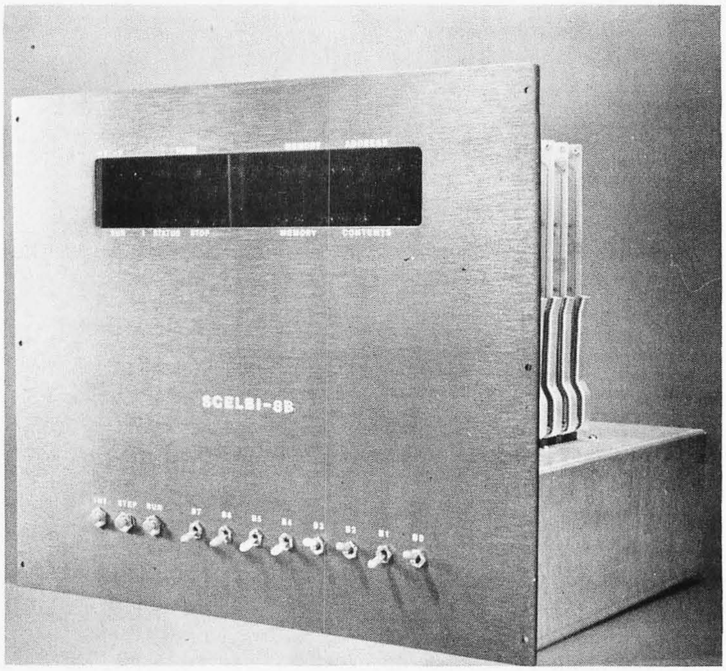
The circuit design techniques for both types of modem are similar. Because of the faster Baud rate, control of group delay distortion in any filtering is extremely important. Detection circuitry must not differentially delay ones with respect to zeros. The Rockwell 10371 Digital Telecommunications Data Interface handles much of the non-filtering aspects of this type of modem. This IC also has a built in UART.

Several additional sources of modem information include the Microdata *Communications Handbook*, *Data Modem Evaluation Guide* by V. V. Villips, and various issues of *Data Communications* and *Telecommunications*.

Fig. 18. Digital discriminator needs no adjustments or calibration.



WHAT SINGLE ELECTRONIC MACHINE CAN BE USED TO PERFORM/CONTROL ALL THE FOLLOWING TYPES OF SERVICES?



- Send morse code
- Control repeater stations
- Operate as a calculator
- Receive/send/buffer data between a wide variety of communication devices
- Monitor instruments
- Control machines
- Sort/compile data
- Test other devices
- Play games

the SCELBI-8B MINI-COMPUTER CAN!

SCELBI COMPUTER CONSULTING, INC. - The company that pioneered in producing the small computer for the individual user with the popular SCELBI–8H, now brings you the new SCELBI–8B with increased capability!

Like the former SCELBI–8H, the SCELBI–8B is built around the amazing '8 0 0 8' "CPU-on-a-Chip" which has been revolutionizing the electronics world.

However, the NEW SCELBI–8B offers extended memory capability at reduced cost! It is directly expandable to 16,384 words of RAM/ROM/PROM memory. This increased memory capability now means the user has the potential in a small and compact computer to support compiler type languages, manipulate sizable data bases for business and scientific applications, and support a wide variety of programs including those that take advantage of external mass memory storage devices.

The NEW SCELBI–8B still retains the outstanding features of its predecessor. Decoding logic for 8 Output and 6 Input Ports is built into the basic computer. Plug-in capability for I/O devices is provided on the chassis. A unique, simple to operate console that utilizes just 11 switches on the front panel makes the SCELBI–8B a pleasure to use.

The NEW SCELBI–8B is backed by a line of low cost SCELBI interfaces which currently include: an interface that turns an oscilloscope into an alphanumeric display system, low cost keyboard and TTY interfaces, and an interface that turns a low cost audio tape cassette into a "Mag–Tape" storage and retrieval unit.

Last, but certainly not least, SCELBI has a wide selection of software ready to run on the NEW SCELBI–8B including: Editors, Assemblers, calculating programs, I/O and general utility routines. Additionally, SCELBI produces publications that can show you how to develop your own custom tailored programs.

The NEW SCELBI–8B is available NOW. (We have been delivering since June!) It is available in three forms. Ultra-low cost "Unpopulated" card sets with chassis kits starting at $259.00*. Complete parts kits for a 1,024 word mini-computer as low as $499.00*. An assembled and tested 4,096 word computer is just $849.00*. Interfaces, accessories, and software sold separately.

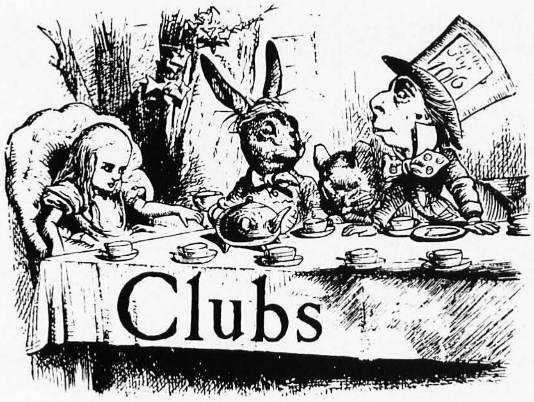
(*Domestic prices.)

(Prices, specifications and availability subject to change without notice)

Literature available for S.A.S.E.

SCELBI COMPUTER CONSULTING INC.

1322 REAR BOSTON POST ROAD
MILFORD, CONNECTICUT 06460



The increasing interest in microcomputers for home and fun and games as well as practical work has led to a number of information centers – the clubs and newsletters organized by readers of BYTE to help promote communications among practitioners of this art. For this first issue of BYTE, I've collected together a "dump" (in English character text, not hexadecimal) of my files on the subject to date.

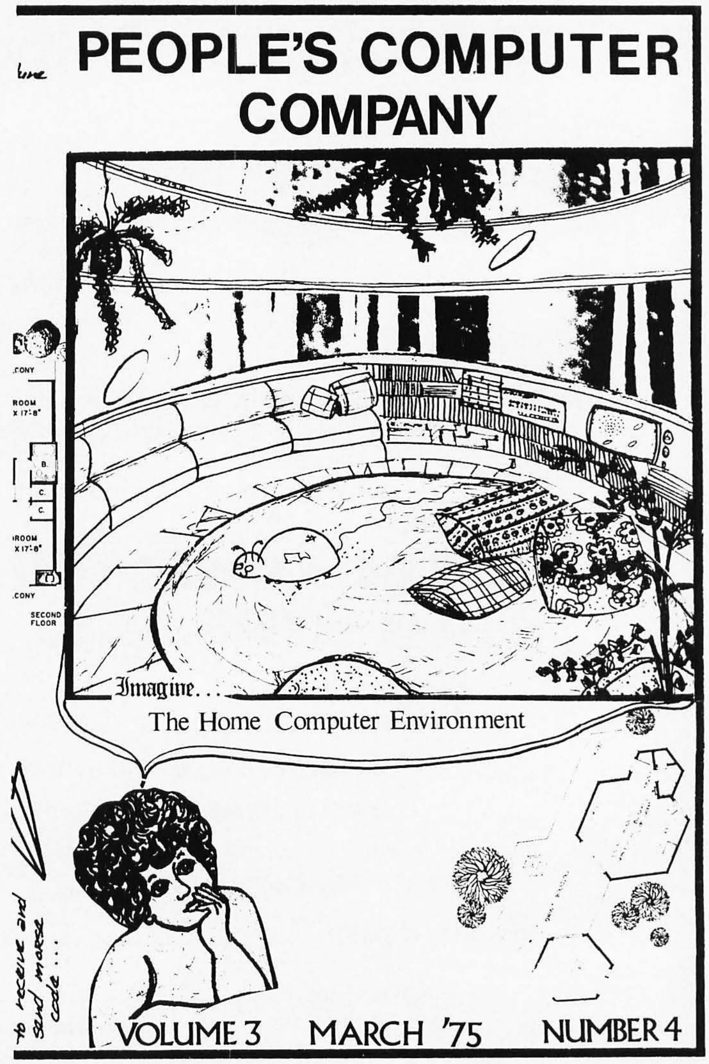
. . . CARL

People's Computer Company
PO Box 310
Menlo Park CA 94025
Editor: Bob Albrecht

This organization puts out a newspaper style publication of information, fantasy, technical designs, etc. It is a "non profit" operation about recreational and educational uses of computers. Publication is on an irregular schedule with subscriptions to 5 issues at $5 to all comers, $3 to those who present some evidence of status as students. The magazine is typeset and assembled with plenty of graphics in what might be called a "neo-Whole Earth Catalog" style. The same organization also runs the PCC bookstore at the same address.

The Computer Hobbyist
Box 295
Cary NC 27511
1-919-467-3145 or
1-919-851-7223 evenings or weekends

The Computer Hobbyist people put out an excellent photo offset newsletter prepared with the help of a microcomputer text editing system. A sample of their product – in the form of an article by Hal Chamberlin comparing three micros – is reprinted by permission in this issue of BYTE. Of particular interest for the coming arguments and controversies over cassette interface methods is their unique method of recording which Hal Chamberlin described to me in phone conversation. The goals of the design are reliability *and* speed independence, which are achieved by a pulse recording technique. Another design published by *TCH* is a fairly sophisticated visual graphics system (it might be called a "Cadillac" among such systems) which uses a highly modified TV and can produce very detailed high resolution pictures. *TCH* also is planning to supply a series of kits with PCs and parts for their designs. *TCH* also is very close to having available a BASIC package for the 8008 computer.



HOMEBREW COMPUTER CLUB NEWSLETTER

Issue number four   Fred Moore, editor, 558 Santa Cruz Ave., Menlo Park, Ca 94025   June 7, 1975

"IT'S A HOBBY"

Yes, a hobby for fun. Interest in home computing is spreading fast. I feel our club is doing a good job in supporting the individual experimenter get his or her system up and flying. There are a lot of obstacles, bugs, and technical tricky problems which can frustrate and discourage a person alone. By sharing our experience and exchanging tips we'advance the state of the art and make low cost home computing possible for more folks.

Bring your beast in for a demo! Can it sing or play games? What tricks does it know? Let's have a look at it.

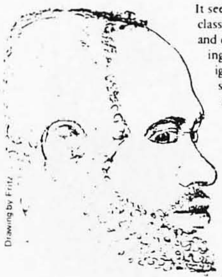
Thanks to Ray and Karen for demonstrating his 008A Microcomputer with audio cassette adapter at the May 14th meeting. Using an 8008 microprocessor, the 008A is available as a kit ($375) from RGS Electronics 3650 Charles St. Suite K, Santa Clara, Ca. Recently Ray got Jerry's TVTypewriter I working nicely.

Thanks to Gordon for bringing and explaining his text editing system May 28th. The system's beauty is the ease with which one can look into 16 K of storage and find what's there to be re-arranged as you please. The only thing I missed in playing REVERSE on it were the bells congratulating me when I'd won. A computer game without bells is like a steam engine without a whistle!

A special thanks to Wayne for bringing the club a paper tape version of a Fortran IV cross assembler for the 8008 and PL/M on mag tape, and two listings of the resident 8080 and 8008 assemblers. The club now has a responsibility to use this software in a non-profit manner, which means no private or commercial deals. If you have a system large enough to house a copy of the mag tape and can make access available to the rest of the members, contact Gordon French.

Wayne also brought a TV terminal Intel developed two years ago as a demonstration unit. The unit uses a 4004 microprocessor and has both a character and a plot mode (5 x 7 dot square you can move around). Wayne hooked it up to a TV and tuned it in on the edge of channel 6. The current ROM gave us our choice of tic-tac-toe or tennis. We played both.

At the previous gathering Wayne had suggested using a shadow ROM for bootstrapping when you first turn on your computer. This time on request he drew a schematic on the greenboard, but I don't think many of the less technically oriented among us followed his explanation completely. Which brings me to a general observation: The club is quite a mixed group. We are composed of outright novices to top flight professionals and leaders in the industry. Many are somewhere inbetween. Only a few are strong in both hardware and software.



Drawing by Fritz

Wayne

It seems to me, we need classes or some more patient and detailed means of conveying information across an ignorance gap, and at the same time not bore the more experienced among us. I think our size is large enough now that after meeting as a whole from say 7 to 9 pm, we then break into three or four small groups for more educationally oriented discussions for an hour. Anyone have comments on this? Perhaps there is enough learning taking place as it is and any attempt to optimize it further will upset the relaxed informality of the gatherings. Comments?

Thanks to John Draper for setting up a group library account for the club at Call Computer. Those who have accounts, have your number changed to a K-2?? number. If we have enough join, the club won't be charged the $5.00 monthly base rate. (We also pay 63 cents per thousand characters on file permonth.) The intention is to have useful programs stored in the K-200 library file.

Thanks to Dan for testing the 2102's the group purchased from Solid State Music. Thanks to Lenny and Frank for setting up the auditorium for our use. Much thanks and appreciation to everyone for your time,energy, and spirit in making the club what it is.

The MITS MOBILE came to Rickey's Hyatt House in Palo Alto June 5th & 6th. The room was packed (150+) with amateurs and experimenters eager to find out about this new electronic toy. The evidence is overwhelming that people want computers, probably for self-entertainment and educational usage. Why did the Big Companies miss this market? They were busy selling overpriced machines to each other (and the government and military). They don't want to sell directly to the public. I'm all in favor of the splash MITS is having with the Altair because it will do three things: (1) force the awakening of other companies to the demand for low-cost computers for use in the home, which will mean competition, resulting in lower prices just as happened with the hand held calculator. (2) cause local computer clubs and hobby groups to form to fill the technical knowledge vacuum. (3) help demystify computers. Computers are not magic. And it is important for the general public to begin to understand the limits of these machines and that humans are responsible for the programming.

Homebrew Computer Club Newsletter
Fred Moore, Editor
558 Santa Cruz Ave.
Menlo Park CA 94025

The Homebrew Computer Club is an organization located in Northern California around Menlo Park. The club was founded by Fred Moore with a hand from Gordon French. A newsletter is published photo offset on a monthly schedule – although no price is quoted, a donation of 50-75¢ per issue would be a fair recompense for costs listed in the club treasury report in issue No. 4. The newsletter has included some excellent design notes by Terry Lee, covering SART chips, power supplies, heat sinking, etc.

Issue No. 4 reports the start of a San Francisco-Berkeley chapter, and refers to another California club:

Sonoma County Minicomputer Club
Mark Robinson, President
1-707-544-2865 (work)
1-707-525-1659 (home)

Amateur Computer Society of New Jersey Is Up and Running

The ACSNJ was first assembled on Friday, June 13, 1975.

The feasibility study was performed by Sal Libes who has become its Operating Manager. The system will run monthly on the second or third Friday of the month.

Input is in the form of 40 + enthusiastic hobbyists. Over 50% of the amateurs are hardware and/or software oriented. There are 10 home computers running in the group, 5 of which are Altairs.

Output will be a local newsletter. The first issue will contain information compiled from a questionnaire given out at the meeting.

Information was processed randomly. There were some minor bugs which had to be worked out; however, those assembled were pleased with the results.

A parts supplier was on hand and welcomed as a local source.

The second running of the ACSNJ was scheduled for Friday, July 18, at the Union County Technical Institute, 1776 Raritan Road, Scotch Plains, New Jersey.

It's good news.

The Amateur Computer Society of New Jersey Is Assembled and Running.

George Fischer
72 So. Railroad Ave.
Staten Island NY 10305

The Amateur Computer Society
Stephen B. Gray
260 Noroton Ave.
Darien CT 06820

Mr. Gray puts out a newsletter. No further information is available about The Amateur Computer Society.



Micro-8 User Group Newsletter
Hal Singer, Editor
Cabrillo Computer Center
Cabrillo High School
Lompoc CA 93436
1-805-733-3501

The "Mark-8" is one of the first widely marketed home computer kits – an 8008-oriented design of Jonathan Titus (TYCHON, Inc., PO Box 242, Blacksburg VA 24060). It first appeared on a large scale in an article in the July 1974 issue of the magazine *Radio Electronics* – and the result of a large response to this product is Hal Singer's formation of the Micro-8 User Group with an initial orientation to the 8008 as implemented via the Mark-8. The newsletter is a self-published offset publication available at $6 for six issues. Much of the information is original as submitted, although Hal summarizes a lot of the stuff with his text editor and printer at the Cabrillo Computer Center. For those interested in the history of one branch of the home computer hobby, the back issues of Micro-8 User Group Newsletter record much activity of the early pioneers of the hobby.

Club in The Dallas-Fort Worth Texas Area

Bill Fuller (2377 Dalworth 157, Grand Prairie, Texas – 1-214-264-0111/1-214-264-9017) organized an informal get together June 29 in a park near Hurst and Bedford, Texas. At that meeting, 12 people appeared – including three Altair owners, the owner of a Martin Research Mark 2 and one home brew purist. Contact Bill for the latest info on activities on the Lone Star state to date.

The Digital Groups
PO Box 6528
Denver CO 80206

This club provides a newsletter of technical and organizational interest which is reproduced photo offset at $6 per year (12 issues). Also offered are kits, boards, and assembled products for miscellaneous peripherals designed by members in the Denver area.

Staccato Notes

Derek McColl reports in phone conversation that he attended the first meeting of an as yet unnamed Los Angeles area computer club. You can reach Derek at 1715 Havemeyer, Redondo Beach CA 90278 to find out about that club.

Was your club or newsletter omitted? No claim is made that this listing is complete. Organizers of clubs are invited to send details of their plans for publication in BYTE.

New England Computer Kibitzers? (NECK) I'll act as an initial focal point for a Boston area computer club. Write BYTE Editorial Offices, Box 378, Belmont MA 02178, or call me at 1-617-729-6914 evenings/weekends.

. . . CARL

COMPUTER-DATA INPUT KEYBOARDS



B5283



B5199

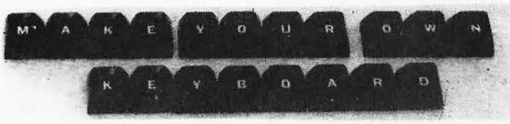
ASCII encoded keyboard. In its own enclosure. Originaly used in SANDERS ASSOCIATES 720 Terminal System. In like new condition. Usefull for any project requiring an ASCii encoded keyboard. 50 Alpha Numeric keys plus 11 computer symbols

STOCK NO.B5283 keyboard $35.00 2/65.00

MICRO–SWITCH (Honeywell) 8 bit binary coded board. 56 keys, alpha - meric and computer symbols Built in TTL decoder. New in factory cartons. A beautiful keyboard.

STOCK NO.B5199 Microswitch keyboard. $45.00 2/80.00

KEYTOPS & SWITCHES TO MAKE YOUR OWN KEYBOARD



We have a large selection of KEYTOPS and SWITCHES, made by RAYTHEON CO. The keytops come in black, grey and white, with contrasting legends. The switches mate with the tops, and are magnetic reed switches. The following combinations are available:

54 key typewriter set, keys only,	black	K9276	2.95
54 key typewriter set, keys only,	grey	K9278	2.95
54 key TTY set, no symbols	white	K9279	2.95
54 key TTY set, with symbols	white	K9282	2.95
54 key set, keys & switches	black	K9288	30.00
54 key set, keys & switches	grey	K9290	30.00
54 TTY set, no symbols keys & Sw.	White	K9291	30.00
54 TTY set, with symbols, keys & Sw.	white	K9291	30.00
11 Key Numeric set. Keys only	Black	K9283	1.50
11 Key Numeric set, Keys only	Grey	K9284	1.50
11 Key Numeric set, Keys only	White	K9295	1.50
12 Key numeric set, Keys only	white	K9286	1.50
11 Key Num.set, keys & switches	Black	K9293	7.00
11 Key Num. set, keys & switches	Grey	K9294	7.00
11 Key Num. set, keys & switches	White	K9295	7.00
12 Key Num. set, keys & switches	white	K9296	7.50
Blank key 1½ keys wide	white	K9297A	3/.25
Blank key 2 keys wide	white	K9297B	3/.25
K9297A with switch	white	K9298A	3/2.00
K9297B with switch	white	K9298B	3/2.00

MINIATURE 7 SEGMENT READOUT

Miniature 7 segment LED readout (EXITON XMN 101). Displays all numbers and 9 letters. O.D. 5/16"x 1/4" Display is .12 ". SPECIAL FOR THIS ISSUE ONLY

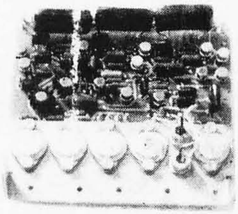
STOCK NO.B5173 with data sheet .50 ea. 5/2.00

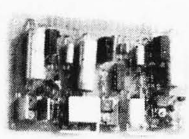
TRANSFORMERS

Computer projects need power supplies. Finding the right power transformer can be a problem. We have one of the largest and most diversified stocks of power transformers in the country. Below we list some representative items in our inventory. Our catalog, free on request lists many more.

36 V. @ 1.0 A. ct, & 6.3 V @ 200 ma.	3 lbs.	B9313	3.50 2/6.00
70 V. @ 1.5 A. ct, & 6.3 V @ 500 ma.	6 lb.	B9314	6.50 2/12.00
90 V. @ 2.0 A. ct, & 6.3 V @ 1.5 A.	8½ lb.	B9315	9.95 2/19.00
50 V.=@ 1.5 A. ct, & 6.3V @500 ma.	6 lb.	B9316	6.50 2/12.00
26 V. @ 1.0 A. ct. &6.3 V. @ 500 ma.	3 lb.	B9318	3.75 2/7.00
38 V. @ 1.5 A. ct. & 6.0 V.@ 500 ma.	2 lb.	B9319	6.95 2/13.00
350 V. @ 35 ma. ct. & 6.3 V. @ 2.7 A	2 lb.	B9321	3.50 2/6.00
70 V. @ 1.5 A. ct. & 6.3 V @ 1.5 A.	7Lb.	B9322	6.75 2/13.00
35 V. @ 6.0 Ct. & 10 V. @ 10.0 A.	6.0 Lb.	B9906	8.95 ea.
64 or 32 V. @ 8.0 A. ct. & 18 V. @ 8.0 A ct.	10 lb.	B9905	11.95

VOLTAGE REGULATOR BOARDS





B5169 is a board containing 3 15 volt high current regulators with 0.1% regulation. 2 of the regulators are rated @ 3 Amps., and the other @ 6.0 amps. The current in each regulator may be doubled with the regulation going to 0.5%.All 3 regulators are short circuit proof, and 2 have electronic crowbar protection. Brand new, in factory boxes.

STOCK NO.B5169 $11.95 ea. 2/21.00

B9013 is a triple regulator with ± 12 volt regulation @ 200 ma. and the third regulator is a tracking regulator, providing regulation from 0 to 5 volta @ .5 A.

STOCK NO.B9013 $5.95 ea. 2/10.00

Both regulators above come withcircuit diagrams.

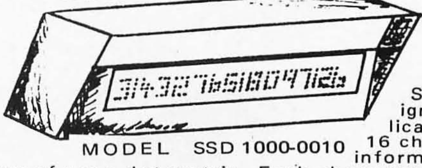
OPERATIONAL AMPLIFIERS

(OP —AMPS)

TYPE	DESCRIPTION	CASE	STOCK	PRICE
709	Hi Performance	TO-5	B4301	.50 5/2.00
4709	Dual 709	DIP	B5301	1.00 6/5.00
741	Hi Performance	DIP	B4316	.65 5/3.00
747	Dual 741	DIP	B4317	1.25 5/5.00
741	Hi Performance	Mini DIP	B4345	.65 5/3.00
747CT	Dual 741	TO-5	B3111	1.25 5/5.00
1458	Dual 741	Mini DIP	B3112	1.25 5/5.00
LM101A	Gen, Purpose	TO-5	B4503	.50 5/2.00

SELF SCAN PANEL DISPLAY

Burroughs



MODEL SSD 1000-0010

BURROUGHS SELF SCAN display, designed for numeric application, requiring up to 16 characters of numeric information. Display is made up of neon dot matrix. Each character is defined by a positive logic 4 bit code. Display operates in a scanning mode, scanning from left to right, one column at a time. Electronics is in interior of bezel, and consists of LSI chip and integrated circuits. Current distributor price is $135.00 . LIMITED QUANTITY

STOCK NO.5180 with data $49.50 2/90.00

Please include sufficient postage. Excess refunded MINIMUM ORDER $5.00

DELTA ELECTRONICS CO.

BOX 1, LYNN, MASSACHUSETTS 01903
Phone (617) 388-4705

Send for the latest edition of our catalog. Loaded with electronic and computer bargains.

WRYTE for BYTE

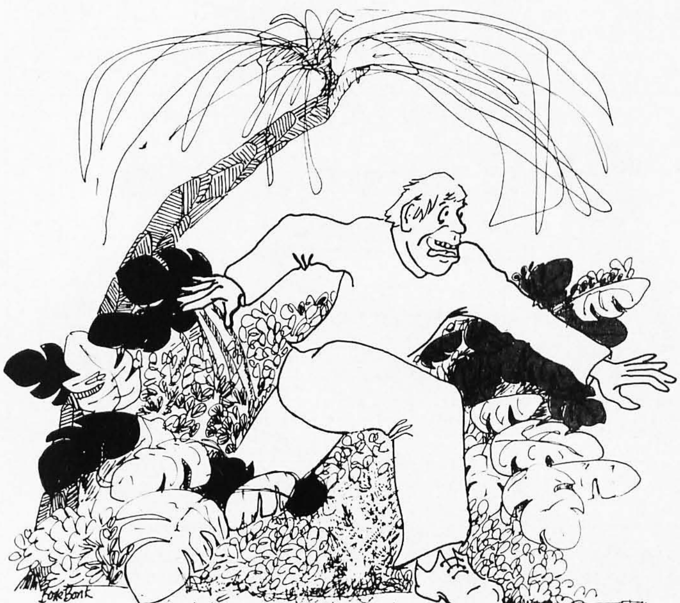
by
Chris Ryland
25 Follen St.
Cambridge MA 02138

As an editor of BYTE, I gave a tour of the author-pitfall jungle to several interested people at a recent (if imaginary) small systems conference. Below is what transpired.

"OK, gentlemen, step right this way; we're heading into the jungle now. Be careful of the pitfall directly to your left. Any questions about the terrain so far?"

"Well, I have an idea for an article, but it's not very original . . . ," says one of the tourists, dropping into the pitfall.

"Oops! I warned him about that! Anyway, the rest of you can benefit from his mistake: that objection just doesn't stand. A new idea might seeem old hat to you, simply because you thought of it. Or, a variation on an old idea can certainly be material for print – what scientist dares claim absolute originality for his research? Furthermore, even small system lore, presented in a tutorial style or approached from a new angle, can be both original and valuable. Look out to your right! More questions?"



"But I can't write anyway . . . " cries another pitfall victim.

"I warned him! That's one of the deepest pitfalls around here. Well, the rest of you probably write much better than you think. It's necessary to have some humility here – your writing may sound bad to you, but no one expects you to win the Pulitzer Prize. Even if you don't write like a pro, think of BYTE as a device for getting good, if rough, ideas into print: submit an article when you've done your 'technical' best, and if we think it's worth it, we'll do our 'editorial' best to get it into publishable shape (leaving your style intact). This is worth emphasizing – most magazines accept only polished articles by professional writers, or else they force their writers into a stylistic straightjacket. But we feel that BYTE can do the most good, and be enjoyable to read, if we publish good ideas in as close to their original form as professionalism permits.

"Well, we've passed through the most obvious dangers, so please be more careful where you walk. Another question?"

"OK, I agree," says an ink-stained tourist, "my idea is good and I can write fairly well. But what good will it do . . ." Another one vanishes into an obvious pitfall.

"I thought you'd all see that one! Your published idea can be invaluable to anyone working on a similar problem. It's unlikely that your work is so unique that no one else could benefit from it! There are also many personal advantages. There's the satisfaction of serving others in your field, and, not least importantly, the money from published articles. Getting your name in print can both give a boost to your professional prestige, and can sharpen your writing ability, a useful tool in any job. Finally, writing about your idea forces you to understand it more completely; also, any feedback from us or from other readers of BYTE can certainly be helpful.

"This brings us nearly to the edge of the jungle. Since it's a lot safer here, you can look back and see where we've been."

"I see! But what can I write about . . .?"

"Darn, I thought I had spotted all the pitfalls! But you can learn from his mistake, and submit feature (medium to long) articles, short articles (tips and techniques, for example), or column contributions; at this point, you might even get a regular column started! As far as 'what to write about?', I'll quote Carl Helmers' ample answer to this question: 'I won't pretend to have a formula answer for that one . . .

" 'Some suggestions of general areas come to mind . . .

" '*Project articles* on new software, hardware or applications designs for systems. Projects which you

have completed or have in progress can be written up for the use of other subscribers. Don't let that "new" intimidate you, either — if it's new to you, and you're enthusiastic about it, then the project is worth setting down on paper for a try at publication.

" '*Special interests.* Do your ideas tend to run along a particular train of thought? Are you a FORTRAN freak, a BASIC fundamentalist, an 8008 hacker, a PDP-8 fanatic, a space war addict, a lover of LIFE, or what have you? Submit an article or "n" to BYTE on your special interest, and it could become an important part of the lore of home brew computing — "bytology" for short.

" '*Surplus equipment.* In Boston, there is a whole fraternity of junk men who often don't know a thing about the stuff they sell, but who sell it at pennies on the dollar. Often such stuff is usable in a computer system, with appropriate knowledge of *how* to use it. I once picked up a $3000 printer for $10 because it didn't look like a printer, and was left in a heap of junk to rot. Write up an article on how to convert particular items of surplus equipment to small systems use, and you will earn the heartfelt thanks of all the other byters who can use your idea. Often the cheapest course to a computer is an appropriately surplus "mainframe" saved from a scrap dealer or found at an auction. But, beware — misadventures can also happen, if you're not careful. Don't be ashamed of your mistakes, though — capitalize — on them by writing up your experience for BYTE.

" '*Games Byters Play.* Fun-type applications make excellent articles. [See the beginning of a series of articles on the Game of LIFE, written by Carl, in this issue of BYTE.] There are other



fun programs to be written using a graphics display or other peripherals with an interactive potential. If you want some ideas, write us, and we'll suggest one or two or two hundred — or, if you have your own ideas, but are puzzled as to how to put them into practice, maybe we can help and the net result would be an article in BYTE.

" '*Practical applications.* Do you also use your byter's system in your business? Many readers are businessmen — doctors, lawyers, architects, engineers, merchants — who justify their expenses by the practical, as well as the fun, applications of their systems. Write an article on business applications — billing, inventory, mailing lists, profit statements, etc. The businessmen in the audience will surely appreciate it. By saving their time and making their businesses more efficient, your article will help improve the commerce that makes all civilized amenities possible.

" '*Education.* Do you have a flair for writing elementary tutorial stuff, with the knowledge to back it up? Write an article or series of articles on the basics. As has been pointed out to me in many letters, the tutorial aspects of design and programming are not to be skimped on — especially if you want to teach your friends and associates so you can talk to them again! It always gives me a great satisfaction to see someone grasp a principle, discover a connection, and experience the delight of knowledge attained. With a published article, although you can't observe this at first hand, the feedback in correspondence should be evidence enough.

" '*Reviews.* Have you built a computer kit? Write an article reviewing your experience with the particular kit. Give the manufacturer an objective treatment — don't blame him for your mistakes — but also be fair to readers by pointing out relative advantages and disadvantages of the product . . . Did you find an interesting book on computer related objects? Review it for your compatriots in the field. Such books include technical works as well as fiction and science fiction along computer lines. (The microcomputer itself is so "science fictiony" that many of my own friends don't really believe in 'em!)

" '*Human interest and creative writing.* Byters appreciate the human aspects of computing. After all, computers are designed, built

The chances are that you know a lot about some aspect of computers — this is your opportunity to write and help other readers. The Pitfall: Waiting for others to write.

Fame (moderate) and Fortune (modest) await your contribution to BYTE.

Send your articles to:

BYTE
Box 378
Belmont MA 02178

Make sure your manuscript is:

- --neatly typed
- --double-spaced
- --one side of paper
- --and includes all drawings, photos, tables and other non-text materials needed.

and used by human beings. There is room for creative writing, humorous anecdotes, and speculations on the evolution of technology, commentary on computing history, etc. Who will be the first to submit an article on the history of Herman Hollerith?

" 'This is by no means an exhaustive list of all the possible topics for BYTE articles. If you don't see your own idea in this list of categories, we can always make a category to fit it into . . . provided you take the step of getting it down on paper . . . . In addition to the standard articles, we will print (without charge, of course) information about club meetings, club organizers, and individuals willing to help others with their home brew systems, in order to foster the growth of the small scale systems idea.'

"Need I say more? And so ends our tour. Now that we're back in civilization, anyone still interested can follow me to the idea bar, where we'll drink a few hints about wryting for BYTE."

Some How To's of Writing a Feature Article

Here is not the place to give an "exposition on the compleat article and its fashioning." But, if you've decided to write a feature article, and if the thought of writing seems to go against every bone in your body, then the following might help. You should feel completely free to approach this task in any way you want; these ideas are only suggestions. But, they *have* worked for many people. Here we go . . .

Outline It

Get your ideas down on paper. Write a few (2 to 5) sentences stating the central idea of your article; this will be your abstract. It should guide the outline — if you ever feel lost, return to the abstract and find where to pick up. Next, the outline. Write down the main section headings, choosing them from the list below, or adding any that are appropriate.

Introduction: Flesh out your idea's skeleton, the abstract, relating the necessary motivation, background, assumptions, and source of ideas for the article. For example, you might tell how the idea came about, what previous BYTE articles your work is based on, and what kind of hardware and software systems it requires for operation.

Overall Design: Discuss and outline the general shape of the system being presented. This should be a "principles of operation" discussion at a relatively high level (but be practical about it), and will usually involve an "overview" of the system components, their actions and interactions. Visual aids such as block diagrams and flow charts are necessities here.

Details of Construction: Whether it be schematics, printed circuit layouts, or program listings, the details are necessary. If they are just too bulky, then this section should cover the system in more detail than the previous section, to whatever level is most helpful.

Construction and Debugging Techniques: The method of construction should be given here, if it is not obvious (and don't assume it is!). Any special techniques or touchy areas should be mentioned, as well as methods of system checkout (give, for example, diagnostic programs or hardware testing instructions).

Operation Instructions: If not given in the overall design section, complete operating instructions should be listed here. Present the system as it appears to a user. Good examples of its use are the most helpful documentation.

Conclusion: Write the inevitable ending section (as short as possible), with mention of possible further developments and applications.

Using the guidelines for the outline headings above, jot down the main ideas under each heading. Take the result, shuffle the headings and ideas until you're satisfied with its structure (note cards with a single item per card are helpful here), and call this your outline. Make a permanent typewritten version of this outline, since it will be your guide in what follows.

Write It

To write an article is to enter a jungle of a different nature than the one we explored earlier. Without actually entering this writing pitfall jungle, I can forewarn you of the most dangerous pitfalls.

"Scribophobia:" This "fear of writing" usually strikes at the outset of the journey. The solution: working directly from the outline, get a first draft written, without stopping to worry about clumsy language or the niceties of grammar. Write without inhibitions, no matter how bad it sounds. The important thing is to get the rough draft written; once it's done, the rest is easy by comparison. Prepare diagrams as you go — they should be written and revised as an integral part of the text, not as a concession to formula.

Straying off the Path: The road to the end of the writing pitfall jungle is rather narrow, but you have a good map: your outline. Straying from it is asking for literary disaster. The written permanence of the outline (did you take my word for it earlier?) should discourage you from changing it too glibly.

Stuffiness: Avoid this danger at all costs; don't confuse technical excellence with highbrow phrases and grammatical "stuffed shirt"-isms. Although this is a matter of style, and would normally be considered in later drafts, the first goes much easier if you write as though talking to someone about your idea. Don't avoid humorous touches if that's your style – they help to keep up reader interest. For example, cartoons that amuse and instruct are a welcome aid. Be intolerant with circumlocutions and useless phraseology. It is clear that phrases like "it is very clear that . . . " are redundant and only drag down the article with dead weight.

Jargon: Be careful of the "in-crowd" approach to writing, which uses jargon and cute phrases unknown to the outsider. On the other hand, don't feel obliged to define common terms or standard abbreviations (but see the later section on article glossaries).

Revise It

Many "How to Write" authors have recommended the fermentation method of revision. Put the freshly-typed (or written) rough draft away, and only come back to it after a few days. This temporal distance gives some objectivity during the next step of revision. This step will usually involve several "passes" over the text, by yourself and hopefully others. Revision involves looking for spelling, technical and grammatical accuracy, logical sequence of ideas, consistency of notation, and completeness of presentation. For example, during revision you may find several out-of-place colloquialisms, a few spelling errors, or an omitted section on a detail of construction. Try and put yourself in the position of a reader who wants to use your idea: can he or she do so with what you've written? Are there any important but unstated assumptions? Any confusing diagrams or descriptions? Aim to be absolutely clear in the technical details of the article.

Shorter Articles

There's really no standard approach to writing shorter articles – the possible range of such articles is so great that it would be difficult to even hint at their "construction." Carl's section on article topics should be a source of some ideas, but my only further advice is this: if you have an idea that's not of feature-article length, then write it in any format you wish, send it in, and we'll try and fit it in somewhere.

Don't Gloss over the Glossary

Since BYTE is aimed at everyone in the small systems world, from junior high school experimenters to industry professionals, a good feature of any article is a glossary. Although it certainly isn't a strict necessity, a glossary should contain the definitions of any words, phrases, notions and abbreviations that might be unfamiliar to a good part of your intended audience. LIFE Line by Carl Helmers, in this issue, is a good example; since his article is (besides other things) a tutorial on system design, and is intended for a wide range of readers, Carl has included definitions of software and hardware terms that might be confusing or "jargon-ish" to the reader. Carl's glossary illustrates that definitions needn't be boring or dry – any sort of information about the terms being defined can be useful, including humorous definitions, anecdotes about derivations or other uses of a term, stories of your previous confusion about some phrase (and how you cleared it up), and so on. If your article contains any kind of "side-light" information about possibly confusing words or concepts, then you should include these "tidbytes" as part of your glossary.

Send It In!

Some practicalities about submission of any kind of article: the text should be typewritten (double-spaced, with ample margins) – if you don't type, it's not expensive to have it done; the point of insertion for each diagram should be clearly marked; the illustrations should be clear and oversized, if they are reasonably simple – our own technical staff can do the final drawings; include any special editorial or publishing instructions in a conspicuous place; if there's a glossary, it should be distinguished from the main body of text in some way. For feature articles, include the abstract and outline in the submission – these items are costless since they are a "spin-off" from a properly-written article. Unusually bulky detailed layouts and listings, although not included in the article, should be submitted, since they can be made available separately.



Probably the worst pitfall is putting off getting started on your article. Procrastination is the thief of fame and fortune, even in the modest amounts offered by BYTE.

THE CURVE TRACER THAT WON'T COLLECT DUST.



The Hickok Model 440 semiconductor curve tracer is all purpose and convenient to use. It's the ideal instrument for testing, evaluating, classifying and matching all types of transistors, FET's and diodes. You'll get stable, full range dynamic displays that you can accurately scale right from the screen.

- Pull-out card for easy, fast set-up and operation.
- Set-up marks for rapid set-up of 80% of tests.
- Unique INSTA-BETA display takes the guesswork out of transistor and FET parameter measurement.
- In-or-out of circuit testing.
- A full range professional tracer at a price you can afford.

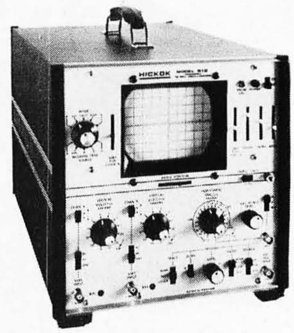
AT YOUR DISTRIBUTOR $165^{00}

HICKOK

the value innovator

INSTRUMENTATION & CONTROLS DIVISION
THE HICKOK ELECTRICAL INSTRUMENT CO.
10514 Dupont Avenue • Cleveland, Ohio 44108
(216) 541-8060 • TWX: 810-421-8286

TROUBLED BY TRIGGERED SCOPES?



The Hickok Model 512 Dual Trace Oscilloscope eliminates the set-up and precision problems you've had to accept using other triggered scopes.

It's easy to set up

- Simplified color-coded front panel controls.
- Beam finder quickly locates off-scale traces.
- Foolproof triggering to 15 MHz.

It gives you superior performance

- 10 MHz response flat within 3dB. Excellent pulse response.
- 3% accuracy on vertical and horizontal ranger.

Hickok industrial lab quality and construction

Glass epoxy PC boards used throughout. Regulated power supply.

AT YOUR DISTRIBUTOR $675^{00}

complete with probes and accessories

HICKOK

the value innovator

INSTRUMENTATION & CONTROLS DIVISION
THE HICKOK ELECTRICAL INSTRUMENT CO.
10514 Dupont Avenue • Cleveland, Ohio 44108
(216) 541-8060 • TWX: 810-421-8286

A PRECISION WAVEFORM GENERATOR AT A PRICE YOU CAN AFFORD.



The Hickok
Model 270 Function Generator gives you a lot more waveform generating capability than you'd expect for its price.

- Puts stable, calibrated, high quality sine, square and triangle waveforms from 1 Hz to 500 kHz at your fingertips.
- With external connections you can produce logic pulses, sweeps and ramps, AM and FM outputs, phase and frequency shift keying signals, tone bursts and more.
- It's an audio generator and much more.

AT YOUR DISTRIBUTOR 166^{00}

HICKOK

the value innovator

INSTRUMENTATION & CONTROLS DIVISION
THE HICKOK ELECTRICAL INSTRUMENT CO.
10514 Dupont Avenue • Cleveland, Ohio 44108
(216) 541-8060 • TWX: 810-421-8286

DIGITAL PERFORMANCE YOU CAN RELY ON.



The Hickok Model 334 DMM is a rugged, non-temperamental, hardworking tool that's easy to use and easy on your eyes. Hickok has established a unique reputation in digital electronics during the past 10 years. The Model 334 is another example of our engineering expertise — an economical lab quality instrument with exceptional durability and accuracy.

- Easy reading, green fluorescent display
- 3½ digit — auto polarity
- 26 ranges including 200 mV AC & DC ranges
- Fast response — 2.5 readings/sec

Basic Accuracies (% of reading)
DC Volts; ±0.2% (±0.5% on 200V, 1200V ranges)
AC Volts; ±0.5% (±2.0% on 200 mV, 2V ranges)
OHMS; ±0.5%
DC Current; ±1.5%
AC Current; ±2.0%

AT YOUR DISTRIBUTOR 229^{00}

HICKOK

the value innovator

INSTRUMENTATION & CONTROLS DIVISION
THE HICKOK ELECTRICAL INSTRUMENT CO.
10514 Dupont Avenue • Cleveland, Ohio 44108
(216) 541-8060 • TWX: 810-421-8286

Write Your Own ASSEMBLER

To date I have not seen any detail descriptions of home brew self assembler systems for microcomputers such as the 8008, 8080, 6800 or PACE. Maybe Dan Fylstra's description of assemblers will start a few readers off in that direction. Dan describes in general terms what assemblers do, scanning techniques, symbol tables, hashing methods and some of the more advanced "bells and whistles" you might employ. Use Dan's article as a source of ideas on the organization and features for your own assembler software designs. . . . CARL

by
Dan Fylstra
14B
550 Memorial Drive
Cambridge
MA 02139

If you have done any work with microcomputers, you have doubtless seen programs written in assembly language. You probably know that assembly language programs must be translated into machine language before they can be executed on the computer. The translation is usually performed by another program, called an "assembler." Because assembly language lets you write mnemonic (easily remembered) names for instructions and data, rather than binary codes, programs may be written more quickly and with fewer errors. The assembler does the tedious job of putting together, or assembling, all of the right bit patterns to make up the program in machine language.

Most of the assemblers presently available for microcomputers are cross-assemblers: They run on big computers or time-sharing systems, and produce output which must be loaded in some way into the microcomputer system. Commercial time-sharing services are expensive, and the whole point of having a home computer is to be able to perform computing chores, such as program assembly, on your own system at ultra-low cost. Since a resident assembler – one which runs on your own micro system – may be unavailable or very costly, you might be interested in writing your own assembler. By doing it yourself you can learn a lot about programming and software design as well as the specs of your own microcomputer, save yourself the cost of program development, and produce a customized language suited to your own needs or fancies. And who knows? – you might even find that other hobbyists or microcomputer users might be willing to pay *you* for a copy of the assembler program that you had so much fun writing.

Now, it's only fair to warn you that writing an assembler is a big undertaking – you'll need a fair amount of time and perhaps some extra RAM chips to accommodate the finished product. But such obstacles have never stopped anyone with your boundless enthusiasm. So the only question is, how do you go about writing an assembler? That's the sort of question that BYTE magazine is designed to answer, and that's what this article is all about.

What Does an Assembler Do?

To answer this question, we have to take a look at some typical machine instructions and how they might be written in assembly language. A machine instruction usually consists of a binary code for some operation, such as addition, and one or more binary numbers denoting the "operands" of the operation. The binary number for an operand may have either of two interpretations: It may denote the binary value of, say, a number, or the ASCII code of a character, or it may denote the binary address of a memory location which holds the actual value of the operand. For example, on the Motorola 6800 the bit pattern

```
1000 1011    0011 0000
 opcode       operand
```

means "add the number 48 (00110000 in binary) to the A accumulator." This might be represented in assembly language as

```
ADDA    #48
```

– note how much more convenient it is to write things this way! In contrast, the bit pattern:

```
1001 1011    0011 0000
 opcode       operand
```

means "add the 8-bit number found in memory location 48 to the A accumulator." This might be written in assembly language as

```
ADDA    BETA
```

where it so happened that, just after the last instruction of a program which was 48 bytes long, the programmer had also written

```
BETA    RMB    1
```

meaning "reserve 1 memory byte at this point, and call it BETA."

These examples illustrate the basic functions of an assembler. In the first case, the instruction's operand was the actual number to be added. (This is often called an "immediate operand.") The assembler read the characters "ADDA" and substituted the proper binary opcode 10001011, and converted the decimal number 48 to its binary equivalent, 00110000. In the second case, the instruction's operand was the address of a memory location. The programmer called this memory location BETA and decided to put it after the instructions of a subroutine, which was to start at location 0. He indicated this by means of the mnemonic RMB, for "reserve memory byte." Since this assembly language statement doesn't actually represent an instruction, but instead tells the assembler what to do, it is often called a "pseudo-op." The assembler read the entire program, counting up the number of bytes that the subroutine would take, and determined that the address of the memory location called BETA was therefore 48, or 00110000 in binary. It assembled this address into the instruction.

The assembler does the tedious job of putting together, or assembling, all of the right bit patterns to make up the program in machine language.

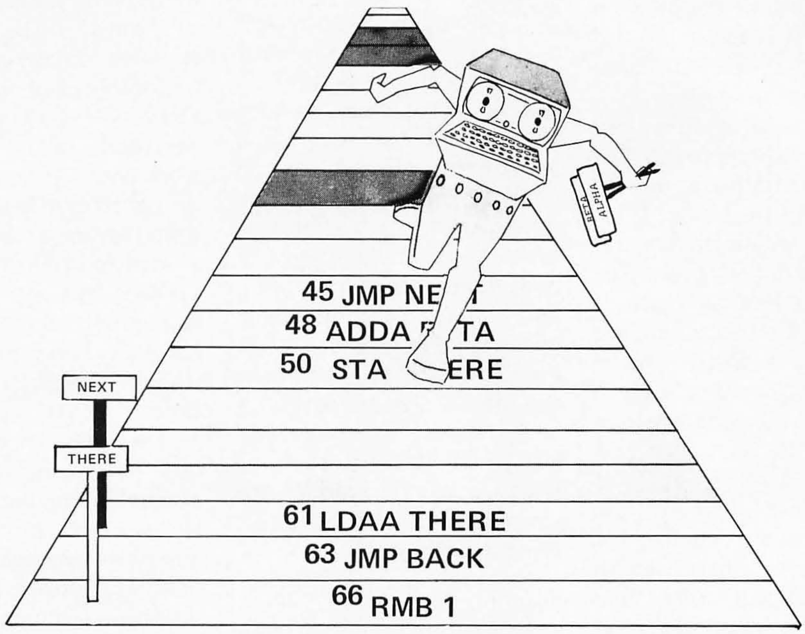
All well and good. But, being an alert reader, you ask, "Wait a minute! What if the ADDA instruction is in the middle of the subroutine? When the assembler reads the ADDA instruction, it doesn't know how many bytes the rest of the subroutine will take, and furthermore it doesn't know (yet) that the memory location BETA is supposed to be reserved just after the subroutine, since it hasn't seen the RMB pseudo-op."

Forward Reference

Fig. 1 illustrates a problem common to all assemblers and compilers, often called the "forward reference problem." There is no neat way out of it. In this case, the programmer could have reserved the memory location BETA before the subroutine rather than after it. But suppose that the subroutine had included a "jump" or "go to" statement:

```
        JMP     NEXT
         .
         .
         .
NEXT    ADDA    #7
```

Fig. 1. The Forward Reference Problem



A two-pass assembler solves the forward reference problem by reading the program twice.

It is rather impractical to try to write every program without any forward jumps!

There are basically two ways to cope with this problem. The first is to read the program once, but to keep sections of the program in memory until all forward references are resolved. Since RAM costs us money in a microcomputer system, we will reject this approach. The second alternative is to read the program twice; an assembler which adopts this strategy is called a two-pass assembler. This approach is slow, but it's also cheap, and that's what we want!

The first time that such an assembler reads the program (i.e., on the first pass), it simply looks at the instruction mnemonics, counts up the number of locations that each instruction will take, and builds a symbol table in memory which lists all of the programmer defined names for memory locations and their corresponding addresses. (We need RAM for this, but not so much as would be required for the first approach.) This process is (somewhat fancifully) illustrated in Fig. 2. Notice that the assembler picks up only the statement labels, ignoring (for the purposes of Pass 1) appearances of the same symbols in the operand fields of instructions.

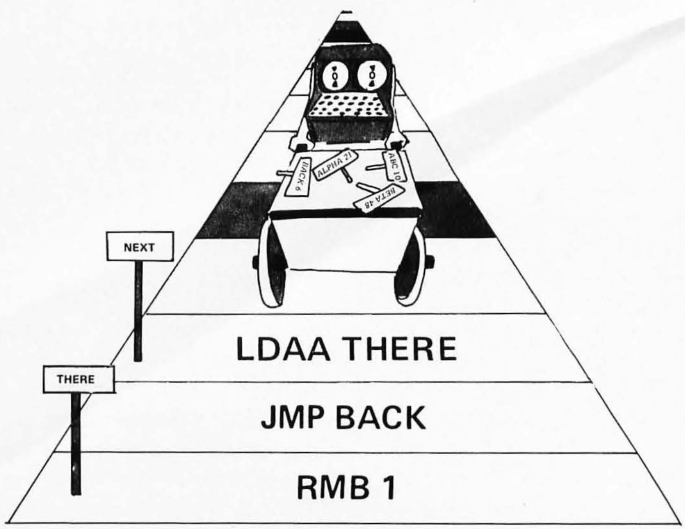
On the second pass, binary opcodes are substituted for the instruction mnemonics, constants are converted to their binary representation, and programmer defined names are replaced by their actual memory addresses, found in the symbol table. This is illustrated in Fig. 3. Any name appearing in the operand field of an instruction which is not already in the symbol table on Pass 2 is undefined in the program, and will cause an error message. One other note: looking up the binary opcode for an instruction mnemonic is essentially the same process as looking up the address for a programmer defined name, so the symbol table can be used for both purposes.

It should be pretty clear by now that an assembler spends most of its time 1) scanning characters, looking for names, numbers and punctuation symbols, and 2) building and searching the symbol table. If we can find simple and efficient ways of performing these operations, and avoid getting them hopelessly intertwined with the rest of the program logic, we should come out with a fairly decent assembler. So let's now take a look at programming techniques for scanning and searching symbol tables.

Scanning Techniques

Our assembler's first task is to scan the characters making up an assembly language program, and find things such as instruction mnemonics, constants and programmer defined names, while noticing but generally ignoring such things as blanks, punctuation symbols and comments. The amateur programmer's first impulse usually is to plunge in by writing a series of tests and branches to handle various

Fig. 2. PASS 1 picks up the labels.



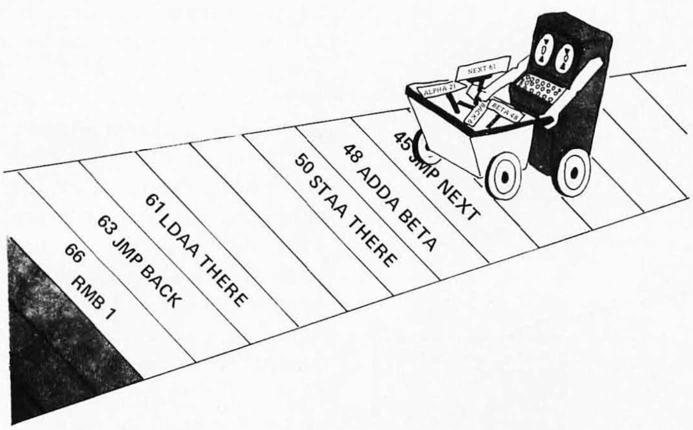
sequences of characters which may appear on a line. This approach frequently leads to the type of scanner known as a "kluge." The computer scientist, on the other hand, has nothing but contempt for this "ill-structured" approach, and prefers to work with regular expressions or right-linear grammars and finite automata. We will take a middle course, outlining some programming techniques that will help make a hand implemented scanner simpler, smaller and faster.

The first technique, if you are designing your own assembly language, is to make it simple to scan! An assembly language statement usually consists of an optional statement label (which then represents the address of the location into which the instruction is assembled), an instruction mnemonic, an operand field, and room for comments. Some assemblers require each element of an assembly language statement to begin in a fixed column or character position of a line, so that the problem of locating the elements for scanning is greatly simplified. However, this is a little rough on the assembly language user, and you will probably save yourself time in the long run by implementing a slightly more complex scanner. To permit a more flexible format, one may take either the "IBM approach," in which a statement label must begin in column 1, an instruction mnemonic must be preceded by at least one blank, and comments are separated from operands by a blank; or the "DEC approach," in which statement labels are followed by a colon (or other punctuation symbol), and comments are preceded by a semicolon. The "DEC approach" is somewhat more

A typical example would be:

EVAL	LDAA	BETA	BEGIN FUNCTION EVALUATION
statement label	instruction mnemonic	operand field	comments

Fig. 3. PASS 2 generates code referencing labels.



convenient and less error-prone for the user, but is slightly harder to analyze. For instance, one must be willing to scan a string of alphameric characters followed by blanks, waiting for a colon or an alphabetic character in order to decide whether the string was a statement label or an instruction mnemonic.

Sometimes a decision as to what to do next must be made on the basis of the type of the next (non-blank) character. If several alternatives are possible, one would like to use a "jump table," or an array of branch addresses indexed by the character code, instead of a sequence of character comparisons. But the ASCII character set allows for 128 different character codes, of which only about 45 are used in assembly language statements. Hence, a common technique for complex scanning problems is to first translate from ASCII to a more convenient set of character codes, using a 128 byte character translation table. The new character codes can be chosen so as to facilitate the use of jump tables at other points.

The elements of an assembly language statement (names, mnemonics and constants) generally consist of variable length character strings, separated by a variable number of blanks. Present-day computers, however, are more adept at handling fixed size objects such as bytes or words. So the most important technique you can use to keep your scanner coherent is to write a "next token" routine, which scans off an alphameric string, a constant (e.g., a string of digits) or a punctuation symbol each time it is called. This routine should return a *code* for the type of item or token just scanned (say, 1 for alphameric strings, 2 for digit strings, 3 for a colon, 4 for a comma, and so on), and a fixed-size *descriptor* giving the address of the first character and the number of characters in the string.

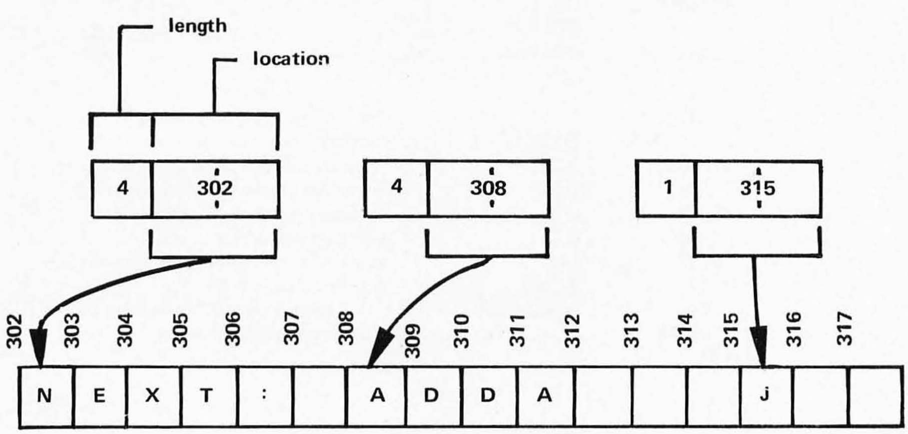
Fig. 4 illustrates descriptors for the statement label, instruction mnemonic, and operand of a typical assembly language statement.

Descriptors for character strings are handy for a number of purposes. Character string move and comparison routines can be written which take two descriptors as arguments. Output lines can be constructed from a sequence of descriptors, and error messages can also be handled in this way. By storing the fixed-size descriptors in the symbol table and the character strings themselves in another area, you can avoid the arbitrary restriction on the length of names to six or eight characters found in many assemblers.

Even more important, the use of a "next token" routine separates the details of scanning individual characters from the problem of deciding how to process each element of a statement. The symbol table routines described below similarly separate the details of identifying particular names and mnemonics from the other problems of processing. These are examples of the use of modularity and hierarchical structure to organize the solution of a complex problem.

Enough in the way of generalizations and philosophy; let's get on with an example to see how all this works. Fig. 5 shows the flow of information from a character code translation routine, to a next token routine, to a routine which determines the type of statement from the instruction mnemonic using a symbol table lookup subroutine. Assembly language for the Intel 8080 has been used in this example. Lower case letters are translated to upper case, and the codes for digits (0-9) and letters (A=10, B=11, . . . ,

Fig. 4. Descriptors Identify Text Tokens in a Line of Characters.



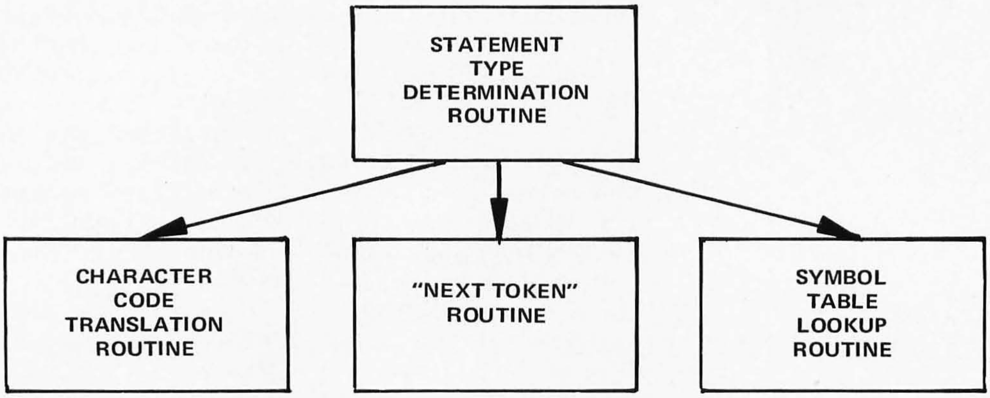
Z=35) are chosen so that alphameric and digit strings can be scanned off using a single comparison for each character. Note the use of a jump table "BRTAB" to select the appropriate handling routine for the next character in the next token routine. Descriptors are returned to the statement type determination routine, and are passed on to the symbol table lookup routine which uses them in character comparisons. The problem of distinguishing statement labels followed by a colon is handled easily at this level: The next token is obtained, and its descriptor is saved; the next token is obtained, and its code is tested; if a colon has been found, the saved descriptor is passed to the symbol table lookup routine, and two more tokens are obtained to balance things out before the instruction mnemonic is processed.

Symbol Tables

The greatest convenience that an assembler provides for the programmer is the ability to give names to memory locations and to refer to those names from other points in the program. The assembler determines the proper address of the memory location, and fills in the address wherever the name is referenced.

The assembler accomplishes this by building a symbol table on its first pass. Each entry of the symbol table contains a programmer defined name in character string form, and the binary address corresponding to it. In addition, the symbol table may contain other character string names, such as the instruction mnemonics or assembler pseudo-ops. The entry for an instruction mnemonic would contain the corresponding binary opcode, and the entry for a pseudo-op might contain the address of a processing routine in the

Fig. 5. Typical (8080) Character Translate and Next Token Routines



CHARACTER TRANSLATION ROUTINE

```
        MVI     H, TABLE    H → page holding table
        LXI     D, LINE     DE → begin of line
        MVI     C, 72       C = length of line
LOOP:   LDAX    D           get next char of line
        MOV     L, A        L = character code index
        MOV     A, M        A = table entry at index
        STAX    D           replace char in line
        INX     D           advance to next char
        DCR     C           reduce no. chars remaining
        JNZ     LOOP        loop for all 72 chars
```

"NEXT TOKEN" ROUTINE

```
        LXI     H, BRTAB    H → branch table base
        LDAX    D           get translated char from line
        RLC                 times 2 for branch table index
        MOV     C, A        set up 16-bit index
        MVI     B, Ø        in registers B and C
        DAD     B           add to branch table base
        PCHL                jump to appropriate routine
BRTAB:  JMP     LETTER
        JMP     DIGIT
        JMP     COLON
        JMP     COMMA
          :
LETTER: XCHG                HL → begin of alpha string
        SHLD    DESCR + 1   put start addr in descriptor
        MVI     A, 36       max translated code for alphameric
        MVI     C, Ø        initialize count of chars in string
SCAN:   INX     H           advance to next character
        INR     C           increase character count
        CMP     M           code < max for Alphanumeric
        JP      SCAN        continue scan if so
        LXI     H, DESCR    HL → length part of descriptor
        MOV     M, C        put in no. chars in string
```

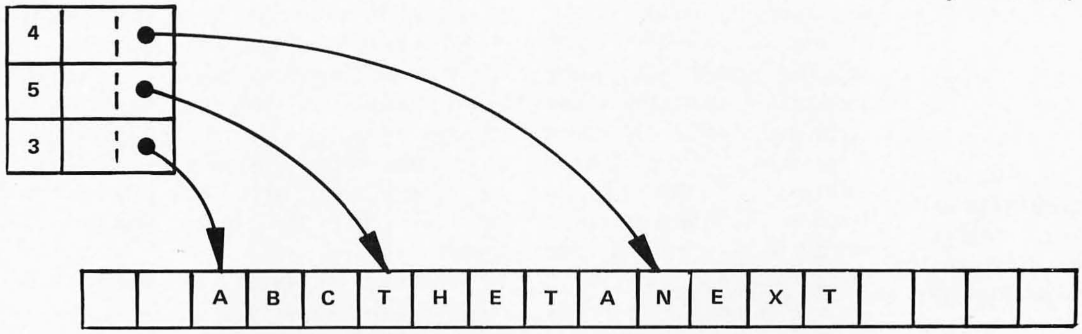


Fig. 6. An Array Symbol Table.

assembler itself. For a computer with several different instruction formats, the entry for an instruction mnemonic might also contain a type code indicating the proper format for this instruction, the number of operands expected, and the interpretation of the operands as addresses or values.

The simplest way of organizing the symbol table would be as an array of descriptors and address words, as illustrated in Fig. 6. Entries are added sequentially to the array during Pass 1. and a sequential search of the whole array is used to find the addresses of programmer defined names during Pass 2. (Each descriptor from the table is passed in turn to a character comparison routine, along with the descriptor for an operand. The comparison fails immediately if the string lengths in the descriptors were unequal.) This type of organization has the great virtue of simplicity, and is probably adequate for a first version of your own assembler. As the programs to be assembled get longer, however, the asembler will spend an increasing fraction of its time searching the symbol table. A faster way of searching the table is needed.

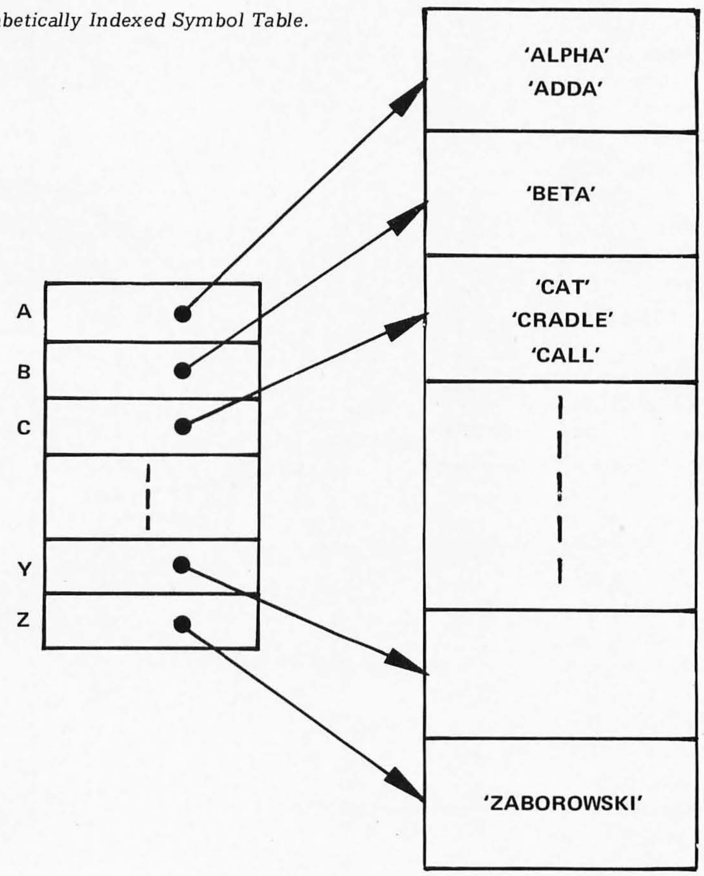
Think about how you would go about such a search, if you were the assembler. What do you do when you open a dictionary or a telephone book? Knowing the order of the alphabet and the thickness of the book, you look at the first character or two of the word, you make a guess at the approximate page, open the book to that page, and begin searching from that point.

Let's have the assembler do the same sort of thing. We will divide up the table into twenty-six sections, one for names beginning with each letter of the alphabet. We know the starting address of each section of the table (we can make a small array of the twenty-six starting addresses), so to look up a name, we look at its first character, go to the appropriate section and search just that section rather than the whole table.

This approach is depicted in Fig. 7.

This is a good first try, but there are some drawbacks. In a program called "assembler," say, you might have a lot of names beginning with A, while in a program called "editor," you might have many names starting with E. On the other hand, your friend Zaborowski might start *all* of his names with Z. Should all of the sections be of the same size? If not, how do you know (at the beginning of an assembly) which sections to make larger? If a section becomes filled, we can simply add the extra names to the next section of the table; now,

Fig. 7. An Alphabetically Indexed Symbol Table.



What happens if we use a random assortment of the names, placing them haphazardly into the various sections of the table?

however, if a name to be looked up on Pass 2 is not found in its original section, most of the following section will have to be searched before the name is found. This phenomenon is called "clustering." Your friend Zaborowski is especially likely to run into this problem, and even if you make the Z section large enough, searching the symbol table will take just as long using the new approach as it did with the old one.

Can we overcome these drawbacks of the new method? Here's where a little lateral thinking will help. We are making use of our knowledge of the ordering of the alphabet. Try the opposite approach: What happens if we use a random assortment of the names, placing them haphazardly into the various sections of the table? At first this sounds absurd, but on closer examination we realize that it *solves the problem!* The problem arose because people are fairly likely to choose a set of names which are related in the alphabetic ordering; by using a randomly chosen ordering, we can minimize the likelihood that a large number of symbols will be placed in a single section of the table. This technique, which is called "hashing" or "hash addressing" for obvious reasons, is used in most modern assemblers and compilers.

So, instead of using the first character of a name to select the proper section of the table, we will use a random assortment of bits, or an arbitrary function of the bit pattern of the entire name, to select a starting point in the table. A function of this sort is called a "hash function". So long as the function's possible values are evenly distributed over the range of addresses for table entries, the problem of the "clustering" or grouping of names will be minimized.

An example of a hash function which usually gives good results is to add together all of the bytes of a character string, ignoring overflow, or else to "exclusive or" the bytes together.

Similarly, in order to minimize the clustering of names which hash to the same starting address, we can "re-hash" the names so as to randomly distribute them around the table. Such a method is called a "random rehash." The following method is easy to implement, efficient, and works well when the table size is a power of 2, say 2**k (see Morris): Suppose a name initially hashes to table entry h, which is already occupied by another name. Initialize a variable R to 1. To rehash the name:

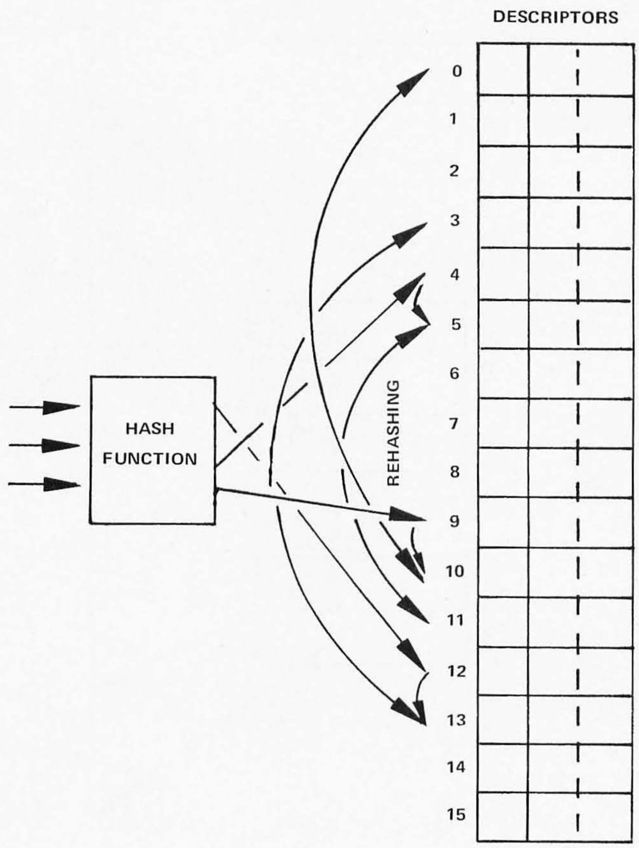
1. Set R = R*5 (shift left two bits, and add to the original number).
2. Mask out all but the low-order k+2 bits of R, and save this as the new R.
3. Shift R right 2 bits and add it to h to get the next table entry h. If this entry is occupied, rehash the name again.

To find a name in the table during Pass 2, we simply hash and rehash in exactly the same way, this time comparing each table entry h against the name to be found. The remarkable fact about this algorithm is that the number of comparisons needed to find an entry, on the average, depends only on *how full* the table is and *not on how large it is.* Even when the table is 90% full, only about 2.56 comparisons will be needed, on the average. In contrast, for a nearly full table of 512 entries, the sequential search method described earlier would take an average of 256 comparisons to find a name, or about 100 times as long!

Random rehashing is illustrated in Fig. 8. The first name to hash to table entry 12, for example, would be stored there, while the next name whose hash function value was 12 would be rehashed to table entry 13, and the next one would be rehashed twice and finally stored in table entry 3.

We have only described one method of hashing here; several other variations are possible. The most important of these is called "hashing with overflow chaining," in which all of the names which hash to the same starting address are chained together on a linked list. This method, which is often used on large computers with dynamic storage allocation, is less suitable for microcomputers because it requires an extra

Fig. 8. Hashing Symbol Table Descriptors.



address field for each symbol table entry. The references at the end of this article can be consulted for a more complete discussion of hashing.

Now that you have become acquainted with some of the basic programming techniques used for scanning and searching symbol tables, you're about ready to start writing your own assembler! You might want to actually try this, using the simplest techniques outlined in this article: Perhaps a fixed-column scanner and a sequentially searched symbol table for a first version. Very often, when it comes to actually getting a program up and running, the simple-minded approach turns out to be the one that works best. Once you've got a basic assembler working, you can consider adding some of the features that we'll discuss next.

More Assembler Features

Up to this point, we have been concerned with only the basic functions of an assembler: The conversion of mnemonics and programmer defined names to instruction opcodes and addresses. Many other features can be added to an assembly language to make it even more convenient for programming. Some of the more useful features of this kind will be considered here.

> **Very often, when it comes to actually getting a program up and running, the simple-minded approach turns out to be the one that works best.**

Defining Constants

Most assembly languages have pseudo-ops which direct the assembler to reserve one or more locations containing constant values. For example, the Motorola 6800 assembly language has a pseudo-op FCB, for "form constant byte." An example of its use would be

```
        FCB     23,$FA
```

which would reserve two bytes containing 00010111 (23 in decimal) and 11111010 (FA in hexadecimal or base sixteen).

Sometimes an instruction takes its operand in a memory location (rather than as an "immediate" operand), but the operand itself is actually a constant. Instead of writing

```
        ADDA    THREE
          .
          .
          .
THREE   FCB     3
```

we would like to be able to write

```
        ADDA    =3
```

and have the assembler automatically reserve a memory location containing 3, and assemble its address into the instruction. Such an instruction operand is called a "literal." On machines where some instructions can address only a limited range of memory locations, this feature may be difficult to implement.

Equivalences

It is often convenient to be able to define a symbol with a constant value, or with the same value as another symbol. For example, a constant representing, say, the size of an array, may be used at several points in a program. By using a symbol in place of the constant throughout the program, and defining the symbol's constant value at the beginning of the program, we can make it easier to change the size of the array when producing a new version: Only the symbol need be redefined, and its new value will be substituted at the appropriate points by the normal process of assembly. (This is called "parameterizing" the program.) This feature is not too difficult to implement, and most assemblers have a pseudo-op such as

```
SIZE    EQU     25
```

which allows us to write

```
        LDAA    #SIZE
          .
          .
          .
ARRAY   RMB     SIZE
```

or, in general to use the symbol SIZE wherever the constant 25 could appear.

Expression Evaluation

Besides defining constants and constant-valued symbols in a program, it is frequently useful to be able to combine such elements into arithmetic expressions whose values can be computed at assembly time, and to use those values in place of other constants. For example, the same program with a parameter SIZE for the size of an array might include statements such as

```
        ADDA    #SIZE-1
          .
          .
          .
SPACE   EQU     3*SIZE+1
```

which would specify (for SIZE=25) that 24 should be added to the A accumulator, and that SPACE should have the constant value 76 wherever it appears in the program.

It is remarkably easy to evaluate expressions of this kind, taking account of parentheses and the normal precedence of arithmetic operations. An algorithm to perform the evaluation of such expressions will be the subject of an article in a later issue of BYTE; if you are impatient, you can consult Mealy or Gries (see the references).

Conditional Assembly

We saw how a program could be parameterized by the use of equivalenced symbols and arithmetic expressions. Sometimes a program can be parameterized in another way: Entire sections of the program can be included or omitted, depending on the values of certain parameters. For example, if the maximum value of a certain variable is less than 256, it can be stored in a single byte on most machines; but if the maximum value is 256 or more, two bytes or a word must be used. Thus we might wish to write something like

```
        .IF     MAXVAL LT 256
ARRAY   RMB     SIZE
        .END
        .IF     MAXVAL GE 256
ARRAY   RMB     2*SIZE
        .END
```

with the intent that, if an earlier EQU pseudo-op had defined MAXVAL as, say, 200, the first RMB statement would be assembled, while if MAXVAL had been defined as, say, 400, the second RMB would be assembled.

This feature is not too difficult to implement, and it is extremely useful. The assembler must simply recognize the .IF and .END pseudo-ops, evaluate the relations, and skip the intervening text on both passes if the relation is false. It is easy to imagine (but somewhat more difficult to implement) extensions to this feature, such as the repetitive assembly of certain program segments.

Macros and Relocation

The most sophisticated assemblers are comparable to compilers in complexity, size and versatility. Some assemblers implement a *macro* facility, which enables the programmer to define new instruction mnemonics which are replaced by parameterized sequences of assembly language statements wherever they appear in the program. When combined with features for conditional assembly, a macro facility provides a powerful tool for extending an assembly language to suit it for a particular application.

We have discussed only absolute assemblers: We began by assuming that the program was to be assembled starting at location 0 (or some other fixed location). When the program is going to be loaded into memory along with other, previously assembled programs, however, we don't know how big the other programs are or in which order they will be loaded. In this case it is necessary to put out *relocation* information along with the assembled program, which says, in effect, "If you load this program at location *m*, you should add the number *m* to the following bytes or words in order to make the addresses come out right." This relocation information is processed by a loader, which is responsible for loading all of the related programs into memory.

While both of these topics are interesting and very important, many pages would be required to do them justice and this article is pretty long already! So we'll content ourselves with the topics already discussed. By this time, you probably have either decided that writing an assembler is too much work, and have stopped reading this article, or else you have found the whole idea very intriguing and are looking forward for the last word. So here it is: Now that you know how to write an assembler, why not get out and give it a try? You have nothing to lose but your innocence about the complexities of system software, and perhaps a little of your time.

Good luck!

Now that you know how to write an assembler, why not get out and give it a try?

References

Barron, D. W. Assemblers and Loaders. *American Elsevier (Computer Monograph Series, No. 6), New York, 1969.*

The most complete, readily available text on the design of assemblers and loaders; also describes one-pass assemblers and meta-assemblers.

Gries, David. Compiler Construction for Digital Computers. *Wiley, New York, 1971.*

A highly recommended text on compiler design: Covers both the theoretical and practical aspects of the problem. Includes a good discussion of hashing and more sophisticated methods of scanning.

Mealy, George. "A Generalized Assembly System (Excerpts)," in Saul Rosen (ed.), Programming Systems and Languages. *McGraw-Hill, New York, 1967.*

A classic paper by one of the pioneers of language translators and operating systems. Presents the idea of descriptors for character strings as well as many other innovations.

Morris, R. "Scatter Storage Techniques," in Communications of the ACM 11:1 *(January 1968), pp. 38-44.*

One of the best general surveys of hashing techniques; includes a good, brief description of hashing with overflow chaining.

TERMS AND ALL THAT STUFF: ADD 50¢ TO ORDERS UNDER $10. CALIFORNIANS ADD TAX. ITEMS IN THIS AD SHIPPED POSTPAID. SORRY, NO COD ORDERS TAKEN.

GODBOUT™

BILL GODBOUT ELECTRONICS
BOX 2355, OAKLAND AIRPORT, CA 94614

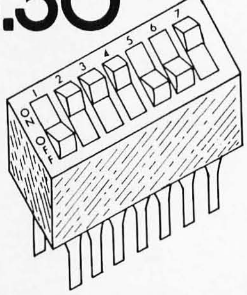
FLYER HYPE: THIS AD SHOWS ONLY THE TIP OF THE ICEBERG. SEND US YOUR NAME, A STAMP, AND YOUR ADDRESS TO RECEIVE OUR FREE FLYER.

NOW...CALL (415) 357-7007, 24 HOURS A DAY, TO PLACE MASTERCHARGE OR BANKAMERICARD ORDERS

4K x 8 Memory Board $130

Several months ago, in conjunction with Solid State Music, we put our our first computer type kit---a 4K by 8 bit memory board. The response has been great; but it's time to move on to newer and bigger things, so we're closing out the ones we have left. Compatible with the MK-8 and other micros; supplied with a 86) pin standard edge connector, sockets for all ICs; features plated-through holes, proper bypassing, flexible addressing, and more. Board is 6½" x 9½".

DIP Switches $1.50



DIP SWITCH -- These programming switches have 7 miniature SPST switches per DIP package. Ideal for computer/digital hobbyists.

Hobbywrap tool $41.95



Thought you couldn't afford to wire wrap? Think again. Cordless for convenient operation. Complete with rechargeable batteries, charger, bit, and instructions; ready to go as soon as the batteries are charged.

ICs

"2102" 1024 X 1 BIT RAM -- SINGLE +5 VOLT SUPPLY; STATIC OPERATION; FAST ACCESS. **$1.95**

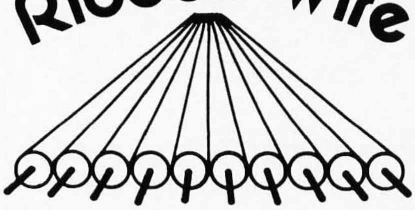
"8008" 8 BIT MICROPROCESSOR -- POPULAR 8 BIT PROCESSOR IC. $27.95

"5203" ERASEABLE ROM -- ORGANIZE AS 4 X 512 OR 8 X 256 BIT MEMORY. STATIC AND NON-VOLATILE; FULLY PROGRAMMABLE AND ERASEABLE. NO CLOCK OR REFRESH REQUIRED. $14.50

ROM programming ⋆now⋆ available

We can program your 5203s, 1702s, and other ROMs for $7.50 a piece, or 10/ $35.00. Call our 24 hour phone line to request hexadecimal coding forms.

Ribbon Wire



This is a flat, multiconductor wire, available in multiples of 10 conductors up to 100 conductors. Comes in 20 ft. lengths only. Cost is 1¢ per conductor foot; for example, a 20 conductor cable 20 ft. long = $4.00

Mag Core Memories

NON-VOLATILE STORAGE

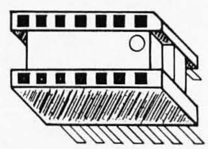
- ---LITHIUM CORE
- ---750 NS FULL CYCLE TIME
- ---350 NS ACCESS TIME
- ---3 WIRES: X & Y DRIVE, SENSE/INHIBIT LINES
- ---DIODE DECODE MATRIX ON BOARD; PRINTS SUPPLIED FOR DRIVE ELECTRONICS AND SENSE AMPS
- ---FOR 8 & 16 BIT MACHINES

☆ stacks ☆

4Kx18 $150

8Kx18 275

IC Sockets



We made a special purchase of PC mount soldertail sockets, including discontinued types and some that need higher than spec insertion force. Brand new from a leading manufacturer of quality IC sockets.

(G)=gold plate contacts (T)=bright tin

T 14 pin	11/$1.95
G 14 pin	10/$1.95
T 24 pin	5/$1.95
T 28 pin	5/$1.95
T 40 pin	5/$2.95
Low profile type sockets:	
G 16 pin	10/$1.95
G 40 pin	5/$2.95

CAVE grafix

GOOD LUCK, BYTE--WE'RE GLAD TO SEE A MAGAZINE FOR THE COMPUTER CROWD!

Tri-share is a trademark of National Semiconductor

"PACE" 16 BIT MICROPROCESSOR

GODBOUT

BILL GODBOUT ELECTRONICS
BOX 2355, OAKLAND AIRPORT, CA 94614

KIT 16 BIT MICROCOMPUTER

WE ARE PROUD TO OFFER THIS EXCITING IC TO THE COMPUTER ENTHUSIAST. IT'S A 40 PIN DIP WITH TRI-SHARE AND A **POWERFUL** INSTRUCTION SET. FAST AND EASY TO PROGRAM; HAS ASSEMBLER & EDITOR READILY AVAILABLE.

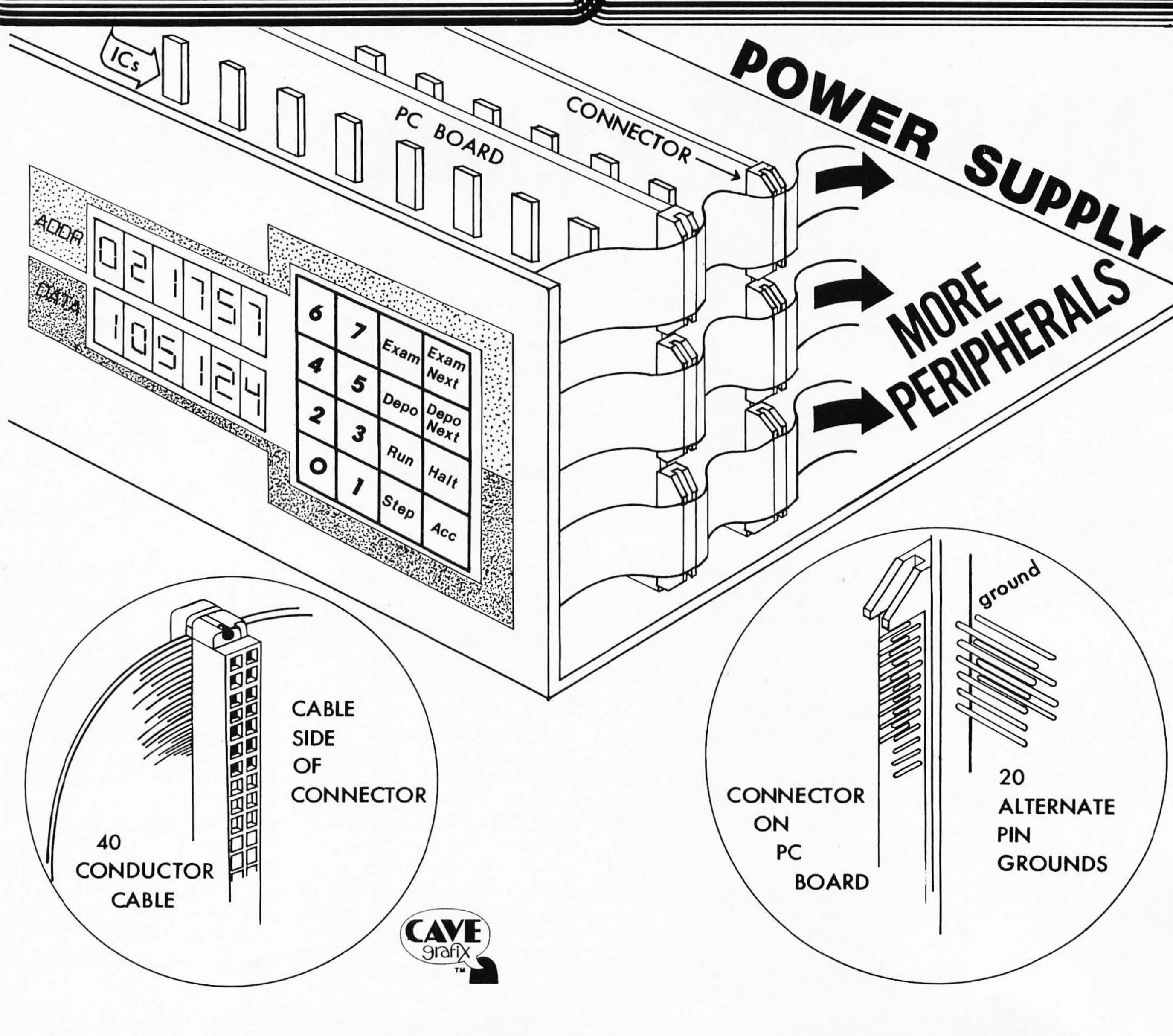
FROM NATIONAL SEMICONDUCTOR, "THE SECRET MICROCOMPUTER COMPANY".

AVAILABLE SEPT. 1 - $125

BELOW IS AN ARTIST'S CONCEPT OF THE MICROCOMPUTER KIT WE'VE WRAPPED AROUND OUR **PACE** CHIP. YOU'LL APPRECIATE THE 7 SEGMENT READOUTS FOR EASY OCTAL DEBUGGING ON BOTH ADDRESS AND DATA, AS WELL AS THE INTEGRAL AUDIO CASSETTE AND SERIAL/TELETYPE INTERFACES...PLUS 2K BYTES OF INTEGRAL MEMORY.

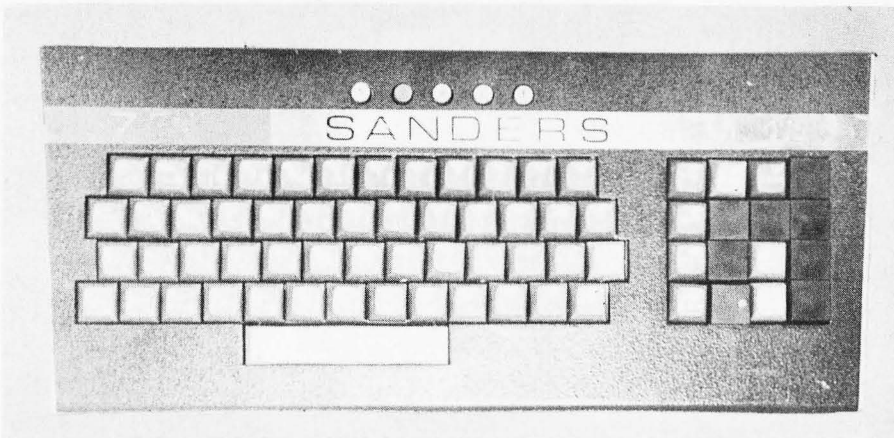
WE DO, HOWEVER, NEED A NAME FOR THE BEAST. SEND US YOUR IDEA---IF WE USE IT, YOU WIN A FREE KIT.

AVAIL. SEPT. 1-Write for details!



CAVE grafix ™

DECIPHERING



MYSTERY KEYBOARDS

Did you ever wonder about the use of surplus keyboards for use in your system? Here is an article describing one way to analyze such a keyboard – illustrated by a particular model which is available through one of BYTE's advertisers. Do you use a surplus keyboard already? This is one of the most common and usable of surplus subsystems – I'd like to see a few reader submitted articles on use of various keyboards available in surplus channels. . . . CARL

by
Carl Helmers
Editor, BYTE

One of the best sources of input data for your home brew computer system is the typewriter style keyboard device. A decent keyboard will give you the ability to enter parallel character data 8 bits at a time. The typical keyboard input devices will also include a flag of some sort to indicate that a key has been pressed. It might also include an "acknowledge" line to be pulsed after the computer had read the data. The parallel interface of a typical keyboard is illustrated in Fig. 1. Fig. 1 is a typical interface of a keyboard, and is used only as a guide to the analysis of an actual keyboard later on in this article.

The manual input of the keyboard is its most important feature. It is the human operator's depression of a selected key which communicates some information to your system. When the key is depressed, it causes the keyboard input device's logic to generate an encoded binary pattern for the key. This encoded binary pattern is typically an ASCII character code presented on the data lines D0 to D6. In addition to the encoding function, the keyboard has logic which produces a "flag" signal to indicate that some key has been depressed. This flag is either a pulse (see timing diagram example in Fig. 1) or a level state, depending upon the particular keyboard design involved. It is often the case (but not required) that the keyboard is designed for interactive control by the computer processor. In such cases, an "acknowledge"

signal must be generated by the computer and sent back to the keyboard to reset the logic of the keyboard input device.

The encoding pattern of the keyboard input device depends upon the manufacturer's design and must be determined for a surplus keyboard before you can use it. For many keyboards, the ASCII pattern of Table I is applicable – each key maps into one of the 7-bit patterns listed. Unless stated by the dealer, you will have to approach the analysis of the surplus keyboard without any assumptions: it is likely to be ASCII but... you could wind up with a Univac "Fieldata" encoded keyboard; you could wind up with an IBM EBCDIC keyboard, etc. Many non-standard encoding schemes for alphanumeric keyboards are derivatives of ASCII. Thus the example in this article is chosen with an ASCII encoding scheme in mind. (IBM surplus is rarely in usable form and the number of EBCDIC keyboards by non-IBM manufacturers is an unknown but assumed small number.) In Table I, the common character codes are shown in a typical graphic form as well as in binary, octal and hexadecimal representations.

Now a new keyboard fully encoded for ASCII and/or EBCDIC is one option you have for implementing a keyboard input device. For example, a new commercial keyboard will typically sell in the $50 to $150 range depending upon options – a keyboard with a standard typewriter style layout and an LSI encoding method. As a second example, Southwest Technical Products Corp. used to sell a hobby quality keyboard at about $40 in kit form. The advantages of new keyboards are obvious: you get the complete description of the hardware along with the product – and an interface which will be similar to the one described in Fig. 1. With the newer LSI encoded boards, you will probably get a keyboard with an "n" key rollover feature to decipher multiple key strokes which overlap in quick succession. This is all well and good, but is there a less expensive alternative? The answer of course is "Yes", and the remainder of this article concerns the techniques involved.

Using Surplus Keyboards

The alternative to new equipment is "pre-owned" equipment, to borrow a term from standard used car dealers' lexicon. Since computers have been in use for a number of years there is a fairly wide selection of equipment in the "surplus" market, as you can find out by reading the advertising pages of BYTE. An item which is frequently found in surplus vendors' offerings is the keyboard input device. Prices for keyboards vary considerably – from $10 for real "junk" to about $40 for premium keyboards. The use you can get out of such a surplus board ranges from a complete subsystem ready to hook up – to a mere array of key switches which must have a new set of encoding logic to make it work.

The keyboards you employ for this purpose must be selected and analyzed on an individual basis – there is no stock formula applicable to all such keyboards. Several rough guidelines will help you keep out of too much trouble:

1. Always look for a unit which is in sound physical condition. Get one which has the cleanest possible key tops, smoothly working keys, little sign of "hack" modifications to PC circuits, etc. Verify that the keyboard is a "switch" type – Hall effect or capacitive keyboards exist and should be avoided without proper documentation.

2. The most desirable keyboard will be one in which the encoding logic is readily decipherable. This will invariably be the case with diode matrix keyboards (see text below) – and may be possible if an LSI chip with a standard part number is utilized.

3. The most desirable keyboard will be one on which the PC layout people have made notations of nice little comments like "+5V", "-12V", "VCC", "A", "$", etc. These are great aids to figuring out the operation of the devices.

If you (at a minimum) satisfy the first criterion above, the keyboard will ultimately be usable, provided it uses actual keyswitches, since you can always construct a switch scanner and/or diode matrix to encode the switches as ASCII binary information.

Diode Matrix Keyboard Analysis – An Example

To illustrate what can be done with surplus keyboards, the remainder of this article concerns the analysis of a particular keyboard input device. The keyboard in question has been advertised recently, and is a fairly typical diode matrix encoded

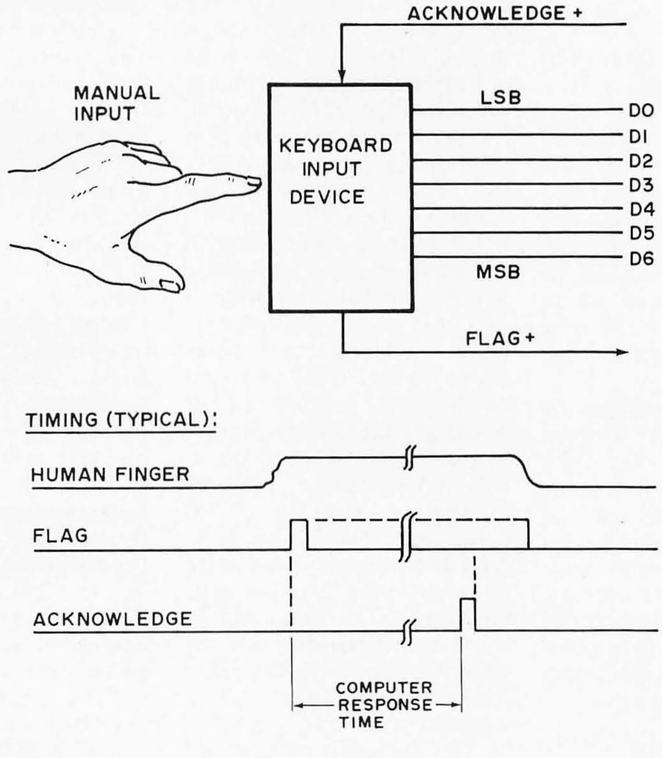
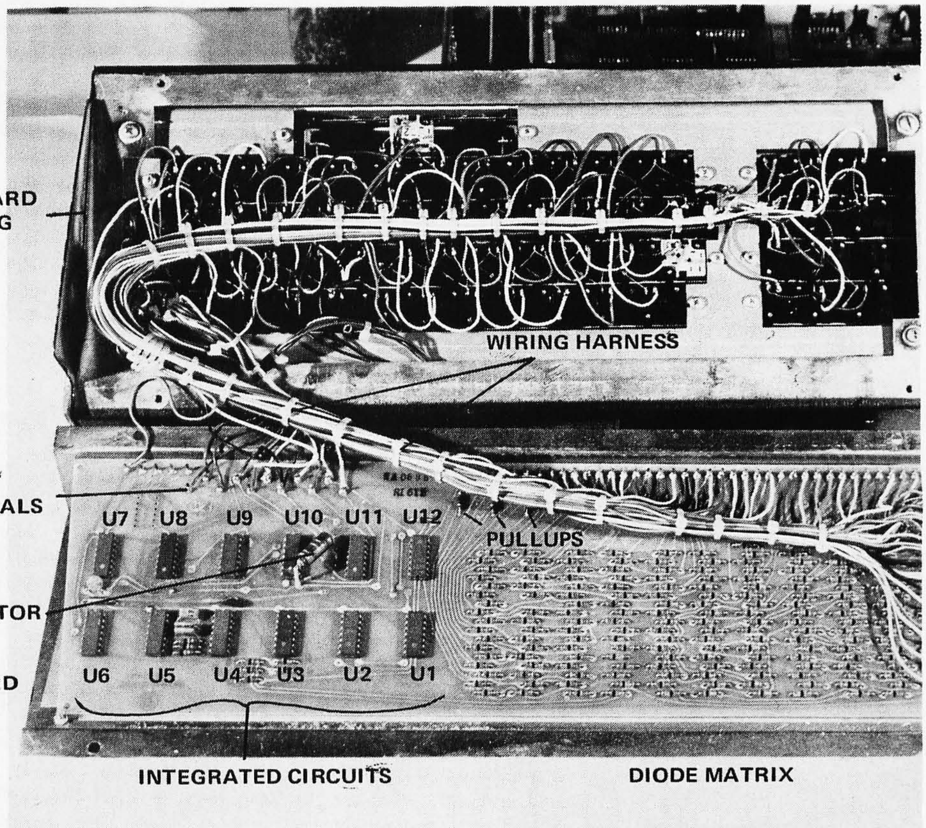


Fig. 1. Typical keyboard functions.



The keyboard, with bottom plate removed and encoder board out in the open. The encoder printed circuit is separated from its mounting on the bottom plate but is still attached by its wiring harness.

keyboard of the 1966-1970 vintage. This keyboard is a surplus Sanders Associates Model 722-1 subsystem, which comes enclosed in a metal housing with a fairly typical Teletype style key layout. On the right hand side of the keyboard is a set of special function keys, which obviously had some meaning in the original system using the device.

The keyboard and housing can be used "as is" in your system – with the only necessary modifications being the substitution of an interface plug and cable which can mate with your own equipment. The example of analyzing and figuring out this keyboard can be used as a guide to similar work with other surplus keyboards.

Start at the Beginning

The object of this project is to determine the details needed to make the Model 722-1 keyboard work – but without any original design documentation from the manufacturer, since it is surplus. The first step is to put on your Sherlock Holmes cap, crank up your deductive powers and begin disassembling the keyboard. In order to analyze the circuit, a likely place to start is the bottom cover plate. In the case of the 722-1, four screws hold the cover plate to the bottom of the housing. Upon opening the cover plate, the 722-1 will be found to have a printed circuit board attached to the plate – a thin plastic sheet glued to the cover plate prevents inadvertent shorting of PC conductors. The PC should be removed from the cover plate by unscrewing the four nuts securing it. The result will be a PC board hanging out the back of the housing/keyboard assembly by its wiring harness.

The actual process of analysis of a keyboard such as this will probably take you an evening or so. The key features to look for in a diode matrix encoder keyboard are identified in the photo.

Keyboard Encoder PC. The typical diode matrix keyboard will have a printed circuit board containing a large number (approximately 100-200) of computer diodes and several integrated circuits, with individual wires running from keyswitches to the PC. Sometimes the functions of encoding and control logic will all be mounted on the same printed circuit as in this example. Occasionally, the logic will be split up into smaller chunks on separate boards.

Wiring Harness. A keyboard is easy to figure out if you can get at it "live" (under power). In this case, a wiring harness allows considerable room for extension so that the key switch matrix and housing can be separated from the encoder board.

Diode Matrix. The way to tell a diode matrix board is by the regular array of diodes found at some point. In this example, the array is at the lower right in the photo. While the array is regular, the actual printed wiring is fairly random – although it will ultimately condense down into a set of bit busses.

Integrated Circuits. This particular keyboard has a bunch of integrated circuits in the left hand portion of the encoder board. The photo illustrates arbitrary reference numbers U1 to U12 for the purposes of this article, since no references were built into the printed circuit board.

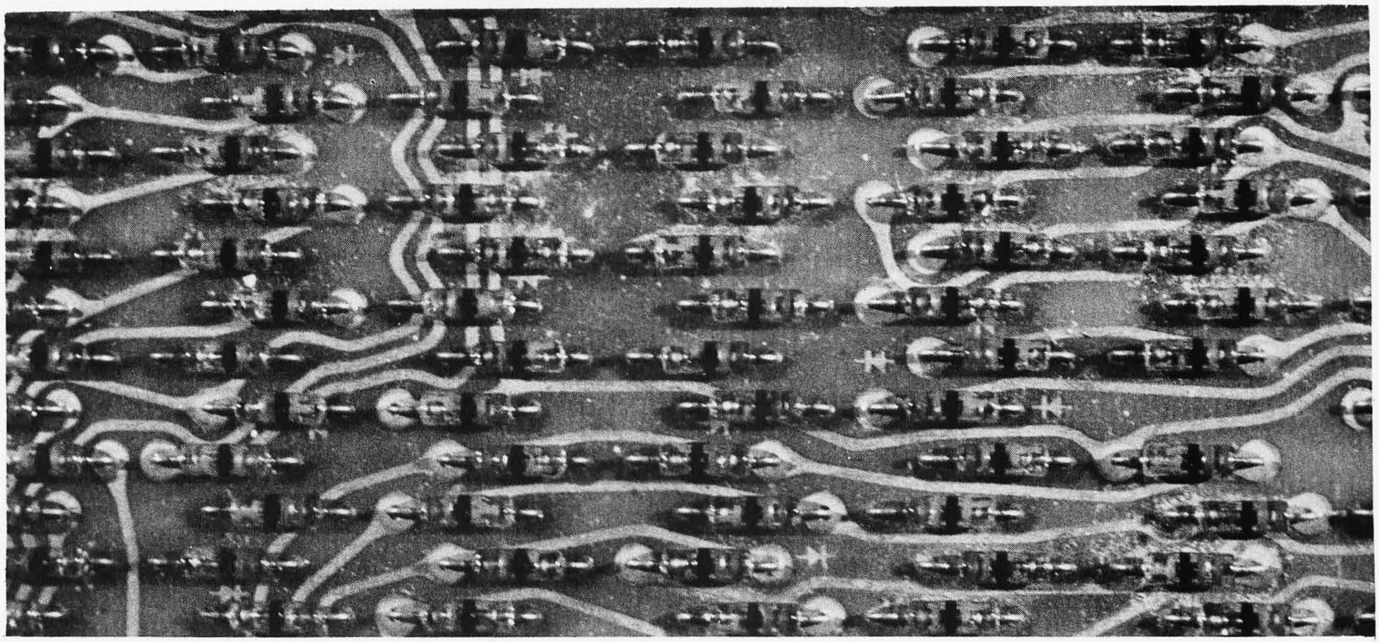
Pullup Resistors. In diode matrix boards, a set of negative logic "wired or" busses is used to generate each bit of the encoded binary word. One pullup resistor (typically 1000 Ohms) is associated with each bus line.

Identifying the Power Requirements

One of the most critical items to be determined in figuring out a keyboard is to identify the power requirements. The best way is

Table I. Binary, Octal and Hexadecimal ASCII Codes. This table contains common symbols for keyboard characters and the corresponding ASCII codes.

Binary	Octal	Hex	Common "Graphics"*	Binary	Octal	Hex	Common "Graphics"*
0000000	000	00	NUL character	1000000	100	40	@ – "at"
0000001	001	01		1000001	101	41	A
0000010	002	02		1000010	102	42	B
0000011	003	03		1000011	103	43	C
0000100	004	04		1000100	104	44	D
0000101	005	05		1000101	105	45	E
0000110	006	06		1000110	106	46	F
0000111	007	07	Bell – Ring the Bell!	1000111	107	47	G
0001000	010	08		1001000	110	48	H
0001001	011	09		1001001	111	49	I
0001010	012	0A	LF – Line Feed	1001010	112	4A	J
0001011	013	0B		1001011	113	4B	K
0001100	014	0C		1001100	114	4C	L
0001101	015	0D	CR – Carriage Return	1001101	115	4D	M
0001110	016	0E		1001110	116	4E	N
0001111	017	0F		1001111	117	4F	O
0010000	020	10		1010000	120	50	P
0010001	021	11		1010001	121	51	Q
0010010	022	12		1010010	122	52	R
0010011	023	13		1010011	123	53	S
0010100	024	14		1010100	124	54	T
0010101	025	15		1010101	125	55	U
0010110	026	16		1010110	126	56	V
0010111	027	17		1010111	127	57	W
0011000	030	18		1011000	130	58	X
0011001	031	19		1011001	131	59	Y
0011010	032	1A		1011010	132	5A	Z
0011011	033	1B	ESC – "Escape"	1011011	133	5B	[– Left bracket
0011100	034	1C		1011100	134	5C	\ – Reverse slash
0011101	035	1D		1011101	135	5D	] – Right bracket
0011110	036	1E		1011110	136	5E	
0011111	037	1F		1011111	137	5F	— Underscore
0100000	040	20	SP – Space	1100000	140	60	
0100001	041	21	! – Exclamation	1100001	141	61	a
0100010	042	22	" – Quotes	1100010	142	62	b
0100011	043	23	# – Number Sign	1100011	143	63	c
0100100	044	24	$ – Dollar Sign	1100100	144	64	d
0100101	045	25	% – Percent	1100101	145	65	e
0100110	046	26	& – Ampersand	1100110	146	66	f
0100111	047	27	' – Apostrophe	1100111	147	67	g
0101000	050	28	(– Left Paren	1101000	150	68	h
0101001	051	29	) – Right Paren	1101001	151	69	i
0101010	052	2A	* – Asterisk	1101010	152	6A	j
0101011	053	2B	+ – Plus sign	1101011	153	6B	k
0101100	054	2C	, – Comma	1101100	154	6C	l
0101101	055	2D	- Minus Sign (hyphen)	1001101	155	6D	m
0101110	056	2E	. – Decimal (period)	1101110	156	6E	n
0101111	057	2F	/ – Slash	1101111	157	6F	o
0110000	060	30	0	1110000	160	70	p
0110001	061	31	1	1110001	161	71	q
0110010	062	32	2	1110010	162	72	r
0110011	063	33	3	1110011	163	73	s
0110100	064	34	4	1110100	164	74	t
0110101	065	35	5	1110101	165	75	u
0110110	066	36	6	1110110	166	76	v
0110111	067	37	7	1110111	167	77	w
0111000	070	38	8	1111000	170	78	x
0111001	071	39	9	1111001	171	79	y
0111010	072	3A	: – Colon	1111010	172	7A	z
0111011	073	3B	; – Semicolon	1111011	173	7B	{ – Left brace
0111100	074	3C	< – Less than	1111100	174	7C	
0111101	075	3D	= – Equality	1111101	175	7D	} – Right brace
0111110	076	3E	> – Greater than	1111110	176	7E	≈
0111111	C77	3F	? – Question Mark	1111111	177	7F	DEL – Delete



Diode matrix – this system of generating the ASCII code is used in older keyboards.

of course to get a keyboard which has power requirements listed on its encoder printed circuit in no uncertain terms. However, "the best" is often a matter of luck and judicious choice of equipment in surplus circles . . . you can make do with less than perfect documentation by employing some knowledge of common design practices. Figuring out power voltages requires the analysis of one circuit power line for each level of voltage involved to completely establish the requirements of the system.

One of the least ambiguous ways to identify power lines is to look up the power pinouts of the integrated circuit components used in your keyboard. This method requires a supply of reference books and a keyboard encoder circuit which uses standard part numbers. For keyboards which are manufactured by the smaller companies in the computer field, parts are usually standard items so that this method can be employed. One of the main justifications for home brew computer clubs is the nice informal arrangement which provides for an exchange of information of this type. In the case of the Sanders keyboard, the two integrated circuit designs used were labelled "ST659A" and "ST680A". The only problem is that no direct reference could be found in literature I had available. However, don't give up with an initial failure to find a reference. What I did after striking out on these two numbers was to look for a similar number differing only in the alphabetical information. I did find references to two DTL integrated circuits "SP659A" and "SP680A," an expandable 4-input NAND gate and a quad 2-input NAND gate. Both these gate designs have package power connections of Pin 8 for power and Pin 1 for ground.

The gate references gave me a high probability determination of the power connections by tracing down ground to the I/0 pin labelled 7 and tracing down power (+5 for DTL) to I/0 pin 5. Being a cautious type of person, I then looked for some independent confirmations of this power pinout identification.

Another method of identifying power and ground connections is to look for color coding on wires. This kind of a confirmation is only possible for boards manufactured with hand wiring. If the harness is one of the multiconductor ribbon cables, color coding is not likely. In the keyboard I analyzed I found that the ground terminal of the decoder was routed via a black wire to the connector on the case, and that the +5 volt terminal was routed to the connector via a red wire. This is consistent with the industry conventions which are used for such wiring – power (positive) is red, ground (negative) is black.

Still another method for determination of power connections is to examine the polarity of electrolytic capacitors mounted on the board for local power supply filtering. These bypass capacitors are often (not always) connected between

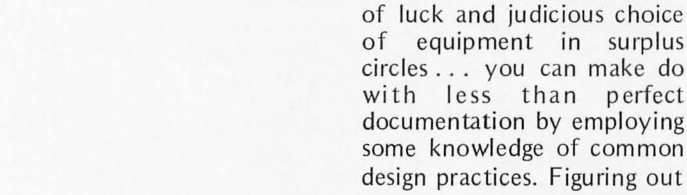
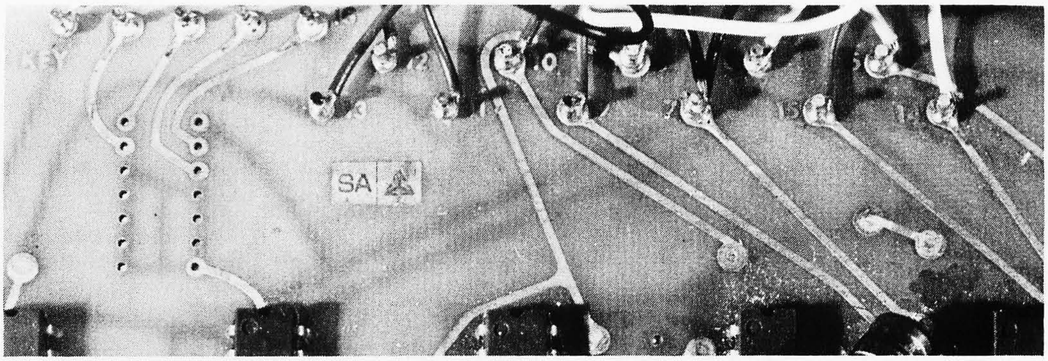


Fig. 2. Turn it on and cross your fingers.



Detail of the output pins. This keyboard is one of the more desirable types – it has labeling of many key features etched along with the printed wiring.

the positive supply and ground, with markings of (+) for the supply side and (-) for the ground side. In the disassembled keyboard photograph accompanying this article, the bypass capacitor is labeled. Using clip leads, the bypass capacitor often provides a handy way to apply power when first testing the board.

Multiple power supply keyboards often occur with later equipment, especially where MOS encoders are employed. This will complicate the analysis problem – often to the point where it might be wise to avoid such boards unless adequately labeled with voltage designations, part numbers and other comments.

Turn It On and Cross Your Fingers?

Now that you think you have the power connections straight, your next step in analysis is to apply a little bit of power to the circuit and see what happens – using a milliammeter. Connect the keyboard using the circuit of Fig. 2. If the power leads have been correctly identified, the current read on the meter should be approximately 100 milliamperes. Remove power ASAP if the meter movement is "pinned" on a 300 or 1000 milliampere scale, since that indicates either a short circuit or incorrect polarity for the power. If a reasonable current (under 300 milliamperes) is drawn, then you can safely trust your power connection determination and proceed by removing the meter from the circuit.

Does It Have a Flag?

The next thing to look for is a "flag" indicating that the keyboard has been activated by a finger and data is present. The term "flag" means a logic line generated in the keyboard encoder which may be either pulsed or steady state. This test requires a method of catching pulses – either an oscilloscope with about 10 MHz bandwidth, or one of a number of logic probes available which "flash" when a state change occurs. Check each of the several I/O connection terminals while pressing a key. If the keyboard is working at all, you will find at least one terminal which changes state – with a pulse or a level change – as keys are activated.

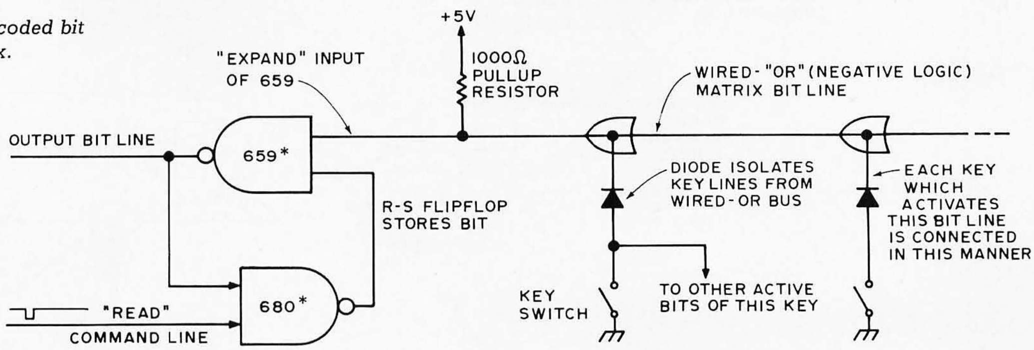
When you have found a pin which changes state, the next test is to see whether it changes the same way for every normal key on the keyboard. If the effects vary from key to key, then the line in question is a data line – if the tentatively identified "flag" pin pulses or changes its level consistently for all keys (with one or two possible exceptions) then it is probably the flag desired. In the Sanders surplus board analyzed here, the flag pin was found to be I/O connection terminal 8.

The exception possible to the "same behavior on every key" statement is evidenced in the Sanders board – the flag interconnection terminal is a pulsed output of 2 microseconds in width for all keys except one: the "Repeat" key causes the flag to change its state. The flag is normally high in this board, but when repeat is depressed it is held low.

Where's the Data?

Now, having found a flag to indicate when data is present, the next problem immediately presents itself – you now turn to examine the other pins of the interconnection to the

Fig. 3. The typical encoded bit line for a diode matrix.



**DTL gates used in surplus "Sanders 720" keyboard; TTL might be used in variations on this theme, e.g.: 7400 series.*

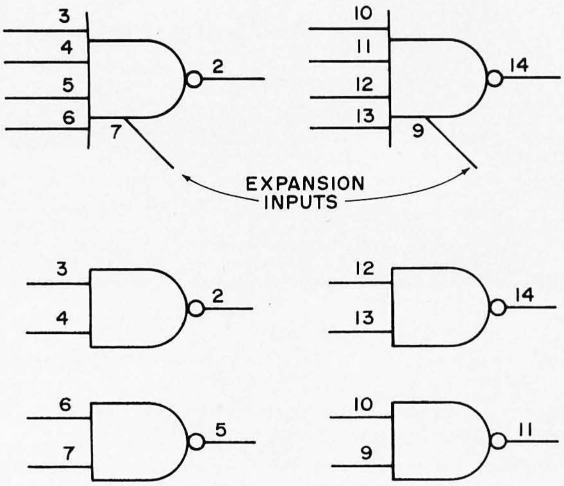


Fig. 4. Pinouts for the DTL gates in the Sanders keyboard. (Unused inputs are assumed logic 1 without external pullups.)

decoder and find no change whatsoever in levels regardless of the key pressed. Ah! The frustration! It's enough to drive you to tracing down the logic of the keyboard, at least for one of the low order data bits. That's exactly what happened in analyzing this example of a keyboard. Fig. 3 is the result of that tracing operation – using the pinouts of Fig. 4 which were obtained from an old (late sixties) data reference for the DTL gates.

As can be seen in Fig. 3, an R-S flip flop is made out of two NAND gate sections for each bit-line of the keyboard. This storage of the state of the diode matrix outputs explains the lack of change seen when first examining the board's outputs for possible data – in order to read (or get ready to read) a key, the R-S flip flops of all diode matrix outputs must be reset. The "read" command line performs this reset. After resetting, the first negative going pulse on the matrix bit line into the 659's expander input sets the flip flop, thus debouncing the contact closure. There is one bit line for each possible bit of "raw data" – and some logic is used to superimpose the shift key and control key information as required.

So, in order to find out which interface terminal corresponds to the "read" command line which resets all the flip flops, a bit more circuit tracing is required. Fig. 5 illustrates the effective logic resulting from the tracing for "Read" – which it turns out is commanded by a negative logic pulse from the computer via interconnection terminal pin 6. In Fig. 5, the R-S flip flop (A) is used to control the computer interface. The receipt of an acknowledge command from the computer resets that flip flop potentially allowing a read, but the NAND gate (B) inhibits recognition of any new keystroke until after the previous key is released. Thus this keyboard has zero-key rollover since all keys must be released before a new key can be recognized.

Figuring Out the Coding

Once the problem of locked up outputs is solved by identifying the "Acknowledge" signal line, the next problem is to identify the bit lines at the interconnection interface. To do this requires the following procedure (by hand) when testing the state of individual bit lines as keys are depressed . . .

1. Short the Acknowledge line to ground.
2. Press a key whose code is to be examined.
3. Look at the outputs on a scope or logic state indicator (the latter is an LED driven by a gate section).

To identify your coding, make the following reasonableness hypothesis initially:

Keys with an identifiable sequential order (eg: alphabetical order) will be consecutive integer numbers in any reasonable binary coding scheme.

You can identify the low order bits in ASCII, for instance, if you make this assumption.

Fig. 5. Keyboard "Read" and Acknowledge logic.

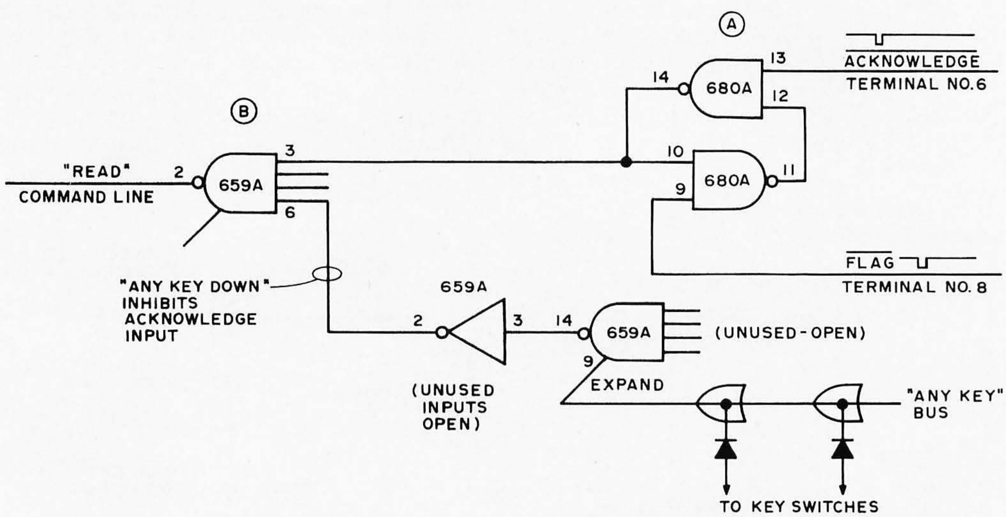


Table II. Terminal Connections for the Sanders surplus keyboard.

Terminal I.D.
#5 Power (+5 volts)
#6 Acknowledge (–)
#7 Ground
#8 Flag (–) (pulse unless REPEAT key held down)
#9 Bit 0 (+) ASCII LSB
#10 Bit 1 (+)
#11 Bit 2 (+)
#12 Bit 3 (+)
#13 Bit 4 (+)
#14 Bit 5 (+)
#15 Bit 6 (+)

So, pick two neighboring keys with identical ASCII high order bits, and test first one then the other (using the three steps above) for each potential bit line until you find a bit line which alternates with your key strokes. Thus, for instance, if you alternately press @ and A on the Sanders board of this article (acknowledging between each look) you will find the state of interface terminal 9 alternating. This can only be the low order bit of the ASCII code. Now pick two keys in alphabetical order which are at a change in bit 1. For example, pick "A" and "B". This will result in all high order bits of the code remaining identical down to the ASCII bit 1 line. Examine the terminals of the encoder while alternately looking at A and B until you find the line which changes.

This procedure can be repeated for the third ASCII low order bit (bit 2) by picking the letters C and D. The bit 2 terminal is found to be 11 by this test for the Sanders board. Continuing once more, test the bit 3 output by looking at G and H alternately (ignore the previously identified pins – all high order pins will remain the same).

By the time terminal 12 is found to be ASCII bit 3, a trend has been established for this keyboard – ascending terminal identifications from 9 are the bits of the ASCII code. In many cases this will be the order of terminals – but you have to identify several of the least significant bits first before you can make a conjecture. This conjecture of ordering can be verified for the Sanders board being analyzed by looking at typical codes (see Table I, and look, for instance, at the output for "line feed" using the terminal identifications listed in Table II).

Now a major input to this identification process is the assumption of ASCII coding – if this assumption gives "funny" results, you have no choice but to use a slightly different method: take each key in turn, depress it, and look at all possible output bits lines of the encode. Record the results in a table similar to Table I, but with the key you find, instead of the standard ASCII. You may find you have inverted data, a completely non-ASCII code set such as EBCDIC, or a modified ASCII.

Now You've Sorted the Bits – So What's Next?

When you have figured out the equivalent of Table II for your own surplus keyboard, the next step is to make a systematic identification in a table similar to Table I. One of the best ways to do this is to use your computer with an input port devoted to the keyboard, and a display or hard copy device for output. A program written to implement the flow chart of Fig. 6 can be used to selectively examine keys on the keyboard. The program accepts a key input, unpacks the bits into a binary and octal form, then displays the bits on your output (TV, character generator, or printer) as a binary and an octal number. If you have a printer output (eg: a Teletype or line printer) then you should write the symbol on the key next to each code after the code is printed. If you only have a display output, then you should note the code on paper along with the key symbol. After you have completed this bit of research, your keyboard is now thoroughly documented so that its input codes can be interpreted by programs. ■

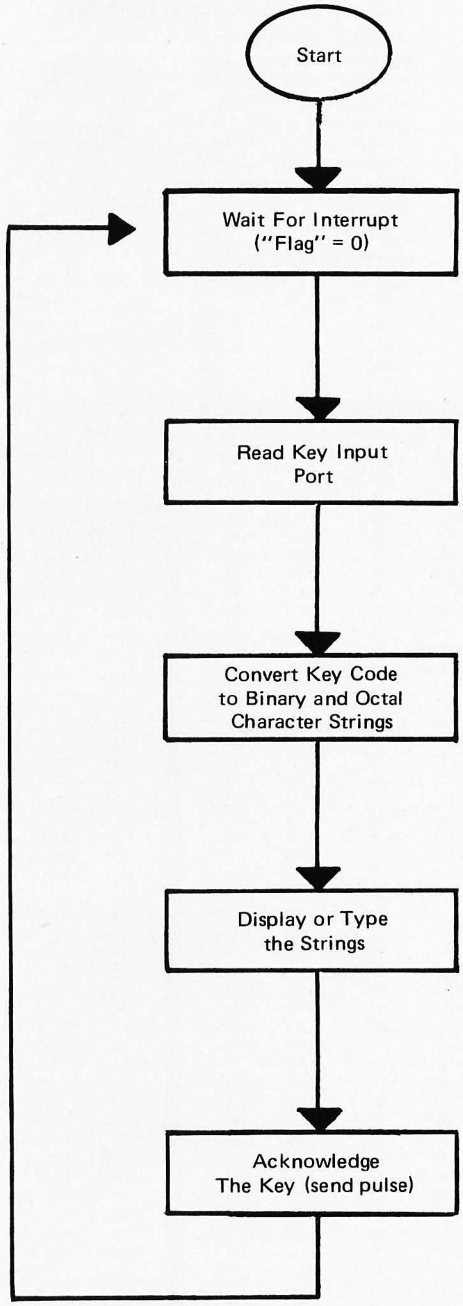


Fig. 6. Keyboard Test Program Flow Chart.

SEPTEMBER SPECIALS

SCHOOL TIME SPECIAL POCKET CALCULATOR KIT

5 function plus constant — addressable memory with individual recall — 8 digit display plus overflow — battery saver — uses standard or rechargeable batteries — all necessary parts in ready to assemble form — instructions included



CALC KIT	**$10.95**
ASSEMBLED	**11.95**
SET OF ALKALINE BATT.	**2.00**

CALCULATOR CHIPS

	#1 Quality	Cosmetic Seconds
CT5001	$1.79 ea.	$ .99
CT5002	2.49 ea.	.99
CT5005	2.99 ea.	1.49

MM5738 8 digit multiplexed — five function — chain operation 2 key memory — floating decimal — independent constant — interfaces with led with only digit driver — 9 V batt. oper. 24 pin **$3.95**

9 DIGIT LED DISPLAY — FNA37



On multiplexed substrate, comm. cathode compatable with all 8 digit calculator chips, 7 segment right hand decimal, red with clear magnifying lens, .12" character, 1 to 4 MA, 1.8 V typ 2⅜" x ¾" x 3/16" high **$2.95**

TTL

7400	$ .13
7404	.15
7408	.15
7410	.13

LINEAR

340T	5V TO-220	$1.25
340T	15V TO-220	1.25
301N	MINI DIP	.15
307H	TO-5	.15

MEMORIES

MM1101	$1.39
MM1103	1.49
MM5203	12.95
MM5261	1.95
MM5262	3.95

TTL

		7491	.97
		7492	.71
7400	$ .14	7493	.60
7401	.16	7494	.94
7402	.15	7495	.79
7403	.16	7496	.79
7404	.19	74100	1.30
7405	.19	74105	.44
7406	.35	74107	.40
7407	.35	74121	.42
7408	.18	74122	.45
7409	.19	74123	.85
7410	.16	74125	.54
7411	.25	74126	.63
7413	.55	74141	1.04
7416	.35	74145	1.04
7417	.35	74150	.97
7420	.16	74151	.79
7422	.26	74153	.99
7423	.29	74154	1.25
7425	.27	74155	1.07
7426	.26	74156	1.07
7427	.29	74157	.99
7430	.20	74158	1.79
7432	.23	74160	1.39
7437	.35	74161	1.25
7438	.35	74162	1.49
7440	.17	74163	1.39
7441	.98	74164	1.59
7442	.77	74165	1.59
7443	.87	74166	1.49
7444	.87	74170	2.30
7445	.89	74173	1.49
7446	.93	74174	1.62
7447	.89	74175	1.39
7448	1.04	74176	.89
7450	.17	74177	.84
7451	.17	74180	.90
7453	.17	74181	2.98
7454	.17	74182	.79
7460	.17	74184	2.29
7464	.35	74185	2.29
7465	.35	74187	5.95
7470	.30	74190	1.35
7472	.30	74191	1.35
7473	.35	74192	1.25
7474	.35	74193	1.19
7475	.57	74194	1.25
7476	.39	74195	.89
7483	.79	74196	1.25
7485	1.10	74197	.89
7486	.40	74198	1.79
7489	2.48	74199	1.79
7490	.59	74200	5.90

Data included with order on request. Add $.30 ea. if item is priced below $1.00 ea. Add $.50 ea. if item is not ordered.

5% OFF ON ORDERS OVER $50.00
10% OFF ON ORDERS OVER $100.00
15% OFF ON ORDERS OVER $250.00

LINEAR CIRCUITS

300	Pos V Reg (super 723)	TO-5	$ .71
301	Hi perf op amp	mDIP	.29
307	Op AMP (super 741)	mDIP	.26
308	Micro Pwr Op Amp	mDIP	.89
309K	5V 1A regulator	TO-3	1.35
310	V Follower Op Amp	mDIP	1.07
311	Hi perf V Comp	mDIP	.95
319	Hi Speed Dual Comp	DIP	1.13
320	Neg Reg 5.2, 12, 15	TO-3	1.04
324	Quad Op Amp	DIP	1.52
339	Quad Comparator	DIP	1.58
340K	Pos. V reg. (5V, 6V, 8V, 12V, 15V, 18V, 24V)	TO-3	1.69
340T	Pos. V reg. (5V, 6V, 8V, 12V, 15V, 18V, 24V)	TO-220	1.49
370	AGC/Squelch AMPL	DIP	.71
372	AF-IF Strip detector	DIP	2.93
373	AM/FM/SSB Strip	DIP	.53
376	Pos. V. Reg	mDIP	2.42
380	2w Audio Amp	DIP	1.13
380-8	.6w Audio Amp	mDIP	1.52
381	Lo Noise Dual preamp	DIP	1.52
550	Prec V Reg	DIP	.89
555	Timer	mDIP	.89
556A	Dual 555 Timer	DIP	1.49
560	Phase Locked Loop	DIP	2.48
562	Phase Locked Loop	DIP	2.48
565	Phase Locked Loop	DIP	2.38
566	Function Gen	mDIP	2.25
567	Tone Decoder	mDIP	2.66
709	Operational AMPL	DIP	.26
710	Hi Speed Volt Comp	DIP	.35
723	V Reg	DIP	.62
739	Dual Hi Perf Op Amp	DIP	1.07
741	Comp Op Amp	mDIP	.32
747	Dual 741 Op Amp	DIP	.71
748	Freq Adj 741	mDIP	.35
1304	FM Mulpx Stereo Demod	DIP	1.07
1307	FM Mulpx Stereo Demod	DIP	.74
1458	Dual Comp Op Amp	mDIP	.62
1800	Stereo multiplexer	DIP	2.48
3900	Quad Amplifier	DIP	.35
8038	V contr. osc	DIP	4.95
8864	9 DIG Led Cath Drvr	DIP	2.25
75150	Dual Line Driver	DIP	1.95
75451	Dual Perepheral Driver	mDIP	.35
75452	Dual Peripheral Driver	mDIP	.35
75453	(351) Dual Periph. Driver	mDIP	.35
75491	Quad Seq Driver for LED	DIP	.71
75492	Hex Digit Driver	DIP	.80

MEMORIES

1101	256 bit RAM MOS	$ 1.50
1103	1024 bit RAM MOS	3.95
2102	1024 bit static RAM	5.55
5203	2048 bit UV eras PROM	17.95
5260	1024 bit RAM	2.49
5261	1024 bit RAM	2.69
5262	2048 bit RAM	5.95
7489	64 bit ROM TTL	2.48
8223	Programmable ROM	3.69
74200	256 bit RAM tri-state	5.90

CALCULATOR & CLOCK CHIPS

5001	12 DIG 4 funct fix dec	$3.45
5002	Same as 5001 exc btry pwr	3.95
5005	12 DIG 4 funct w/mem	4.95
MM5725	8 DIG 4 funct chain & dec	1.98
MM5736	18 pin 6 DIG 4 funct	4.45
MM5738	8 DIG 5 funct K & mem	5.35
MM5739	9 DIG 4 funct (btry sur)	5.35
MM5311	28 pin BCD 6 dig mux	4.45
MM5312	24 pin 1 pps BCD 4 dig mux	3.95
MM5313	28 pin 1 pps BCD 6 dig mux	4.45
MM5314	24 pin 6 dig mux	4.45
MM5316	40 pin alarm 4 dig	5.39

LED'S

MV10B	Red TO 18	$ .22
MV50	Axial leads	.18
MV5020	Jumbo Vis. Red (Red Dome)	.22
	Jumbo Vis. Red (Clear Dome)	.22
ME4	Infra red diff. dome	.54
MAN1	Red 7 seg. .270"	2.19
MAN2	Red alpha num .32"	4.39
MAN4	Red 7 seg. .190"	1.95
MAN5	Green 7 seg. .270"	3.45
MAN6	.6" high solid seq.	4.25
MAN7	Red 7 seq. .270"	1.19
MAN8	Yellow 7 seq. .270"	3.45
MAN64	.4" high solid seq.	2.95
MAN66	.6" high spaced seq.	3.75
MCT2	Opto-iso transistor	.61

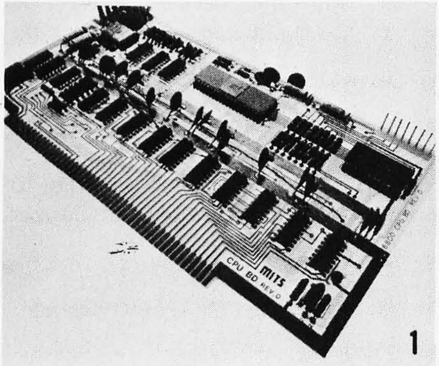
CMOS

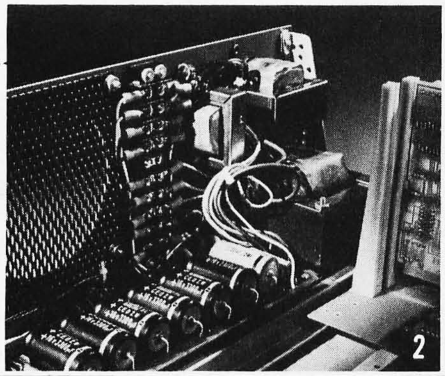
		4015A	1.49	4049A	.59
		4016A	.56	4050A	.59
4000A	$ .26	4017A	1.19	4066A	.89
4001A	.25	4020A	1.49	4068A	.44
4002A	.25	4021A	1.39	4069A	.44
4006A	1.35	4022A	1.10	4071A	.26
4007A	.26	4023A	.25	4072A	.35
4008A	1.79	4024A	.89	4073A	.39
4009A	.57	4025A	.25	4075A	.39
4010A	.54	4027A	.59	4078A	.39
4011A	.29	4028A	.98	4081A	.26
4012A	.25	4030A	.44	4082A	.35
4013A	.45	4035A	1.27	4528A	1.60
4014A	1.49	4042A	1.47	4585A	2.10

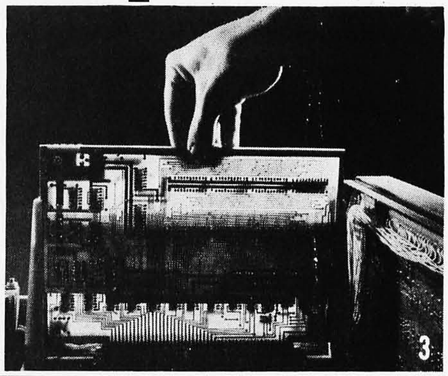
Satisfaction guaranteed. Shipment will be made via first class mail within 3 days from receipt of order. Add $.50 to cover shipping and handling for orders under $25.00. Minimum order $5.00. California residents add sales tax.

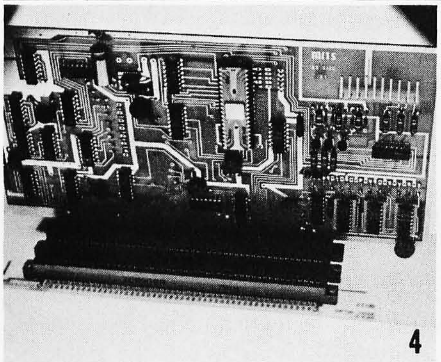
INTERNATIONAL ELECTRONICS UNLIMITED
P.O. BOX 1708/ MONTEREY, CA. 93940 USA
PHONE **(408) 659-3171**

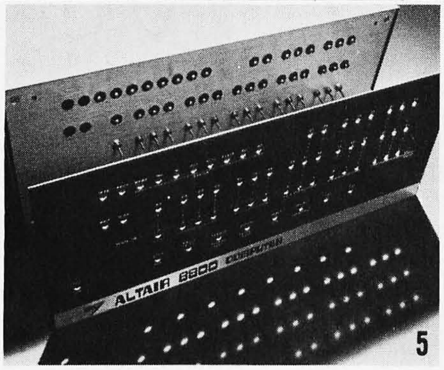
INSIDE the Altair Computer

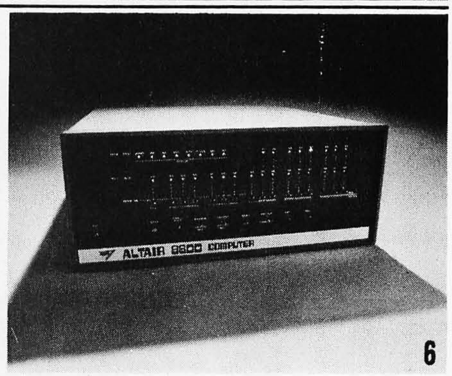

1


2


3


4


5


6

1. Central Processing Unit (CPU) Board. This double-sided board is the heart of the Altair. It was designed around the powerful Intel 8080 microprocessor—a complete central processing unit on a single LSI chip using n-channel silicon gate MOS technology. The CPU Board also contains the Altair System Clock—a standard TTL oscillator with a 2.000 MHz crystal as the feedback element.

2. Power Supply. The Altair Power Supply provides two +8, a +16 and a −16 volts. These voltages are unregulated until they reach the individual boards (CPU, Front Panel, Memory, I/O, etc.). Each board has all the necessary regulation for its own operation.
The Altair Power Supply allows you to expand your computer by adding up to 16 boards inside the main case. Provisions for the addition of a cooling fan are part of the Altair design.

3. Expandability and custom designing. The Altair has been designed to be easily expanded and easily adapted to thousands of applications. The basic Altair comes with one expander board capable of holding four vertical boards. Three additional expander boards can be added inside the main case.

4. Altair Options. Memory boards now available include a 256 word memory board (expandable to 1024 words), a complete 1024 word memory board, and a 4,096 word memory board. Interface boards include a parallel board and 3 serial boards (RS232, TTL and teletype). Interface boards allow you to connect the Altair Computer to computer terminals, teletypes, line printers, plotters, and other devices.

Other Altair Options include additional expander boards, computer terminals, audio-cassette interface board, line printers, ASCII keyboards, floppy disc system, alpha-numeric display and more.

5. All aluminum case and dress panel. The Altair Computer has been designed both for the hobbyist and for industrial use. It comes in an all aluminum case complete with sub-panel and dress panel.

6. It all adds up to one fantastic computer. The Altair is comparable to mini-computers costing 10-20 thousand dollars. It can be connected to 256 input/output devices and can directly address up to 65,000 words of memory. It has over 200 machine instructions and a cycle time of 2 microseconds.
You can order the Altair Computer by simply filling out the coupon in this ad or by calling us at 505/265-7553. Or you can ask for free technical consultation or for one of our free Altair System Catalogues.

PRICES:

Altair Computer kit with complete assembly instructions $439.00
Assembled and tested Altair Computer $621.00
1,024 word memory board .... $97 kit and $139 assembled
4,096 word memory board .... $264.00 kit and $338.00 assembled.
Full Parallel Interface board .... $92.00 kit and $114.00 assembled.
Serial Interface board (RS232) .... $119.00 kit and $138.00 assembled.
Serial Interface board (TTL or teletype) .... $124.00 kit and $146.00 assembled

NOTE: Altair Computers come with complete documentation and operating instructions. Altair customers receive software and general computer information through free membership to the Altair User's Club. Software now available includes a resident assembler, system monitor, text editor and BASIC language.

MITS/6328 Linn NE, Albuquerque, NM, 87108 505/265-7553

MITS

"Creative Electronics"

Prices and specifications subject to change without notice. Warranty: 90 days on parts for kits and 90 days on parts and labor for assembled units.

MAIL THIS COUPON TODAY!

☐ Enclosed is check for $________
☐ BankAmericard #________
☐ or Master Charge #________
☐ Credit Card Expiration Date ________
☐ Altair Computer ☐ Kit ☐ Assembled
☐ Options (list on separate sheet)
Include $8.00 for postage and handling.
☐ **PLEASE SEND FREE ALTAIR SYSTEM CATALOGUE**

NAME________
ADDRESS________
CITY________ STATE & ZIP________
MITS/6328 Linn NE, Albuquerque, NM, 87108
505/265-7553

LIFE *Line*

by
Carl Helmers
Editor, BYTE

Games played with computer equipment are applications of value above and beyond the momentary "hack" value of putting together an interesting program. The creation of a game is one of the best ways to learn about the art and technique of programming with real hardware and software systems. LIFE Line concerns a game — the Game of LIFE, originated by Charles Conway and first publicized by Martin Gardner in *Scientific American*. The Game of LIFE serves as the central theme of LIFE Line — a well defined application of the type of hardware and software which is within the reach of BYTE readers. The description of the LIFE application is the "down to earth" goal of LIFE Line. However, I have an ulterior motive as well — LIFE Line is a very convenient and practical vehicle for teaching ideas about program and system design which you can apply for your own use. Even if you never implement a graphics output device and interactive input keyboards, you can gain knowledge and improve your skills by reading and reflecting upon the points to be made in LIFE Line. The LIFE application also has the side benefit of illustrating some techniques of interactive visual graphics which can be used much more generally.

The Starting Point

In developing a system, it always helps to know *what you want to do!* The ability to pin down a goal for a programming effort — indeed, any effort you make — is one of the most important tools of thought you have available (or can develop) in your personal "bag of tricks." Goal setting does not necessarily mean a complete and detailed description of the result — the feedback from the process of reaching the goal can often modify the details. Goal setting means the setting of a standard in your mind — and on paper — of what you want to accomplish. This standard is used to evaluate and choose among alternatives in a methodical approach to a system which meets that standard.

How to Get From Here to There

The goal of LIFE Line is a hardware/software system which enables the home brew computer builder such as you or me (the "byter") to automate the game of LIFE using relatively inexpensive equipment. It's appropriate here to give a preliminary road map of the course LIFE Line will take, as an illustration of the first steps in the development of a complicated system . . .

1. *The facts of LIFE.* Defining the rules of the game and its logical requirements always helps — after all, I would not want to confuse it with chess, poker or space war!

2. *What do I need to implement LIFE?* Once I know the rules, my next problem is to sketch the hardware and software requirements for a reasonable implementation.

3. *Programming.* Given the necessary hardware, the biggest lump of effort is the process of programming the application. Some parts of this lump include . . .

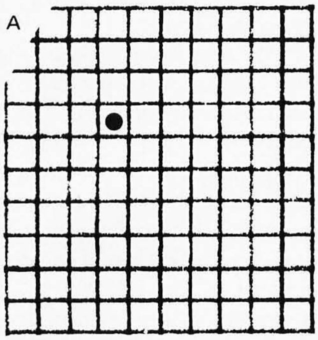
   —*Control flow:* Outlining the major pieces of the program and their relationships.

   —*Partitioning:* A well designed system is simple! But how can the desired simplicity be reconciled with "doing a lot." One way is to partition the system into pieces. Within each piece, a further partition provides a set of sub-pieces and so on. Each piece of the program is thus kept at a level of relative simplicity, yet the whole system adds up to a quite sophisticated set of functions.

   —*Coding:* With the application design laid out in some detail, the program must be coded and debugged for a particular computer. The result could be a series of octal or hexadecimal numbers for your own computer, or a high level language program which can be translated by an appropriate compiler.

Fig. 1. Three views of LIFE: (a) on paper; (b) in memory; (c) on a display.

A



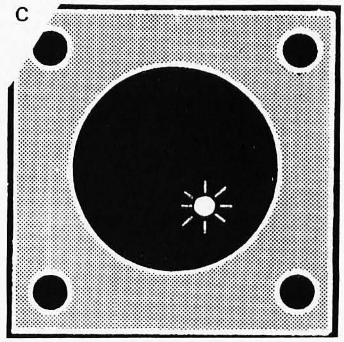
A live "cell" is a dot on paper.

B

```
      0000000000000
     00000000000000
    000000000000000
   0000000000000000
   0000000000010000
   0000000000000000
   0000000000000000
   0000000000000000
   0000000000000000
```

A live "cell" is a "1" bit in memory.

C



A live "cell" is a point of light on a graphics display.

What Are The Facts of LIFE?

Ask a biologist the question "What are the facts of life?" and you will get one answer; ask a "byter" and you'll get the "real" answer – an evolution algorithm used to generate the placement and "cell" content of a square grid given the previous state of cells in the grid. The inspiration of the game is a combination of modern biology, the concept of "cellular automata" in computer science and the pure fun of mathematical abstractions. In making a computer version of the game, the simplest approach is to think of a group of individual "bits" in the computer memory – with your thoughts assigning one memory bit to each "square" of the grid. (The hand operated form of the game algorithm uses graph paper for the squares in question.) If I have a place in memory which can store one bit, it can have a value of logical "zero" or logical "one".

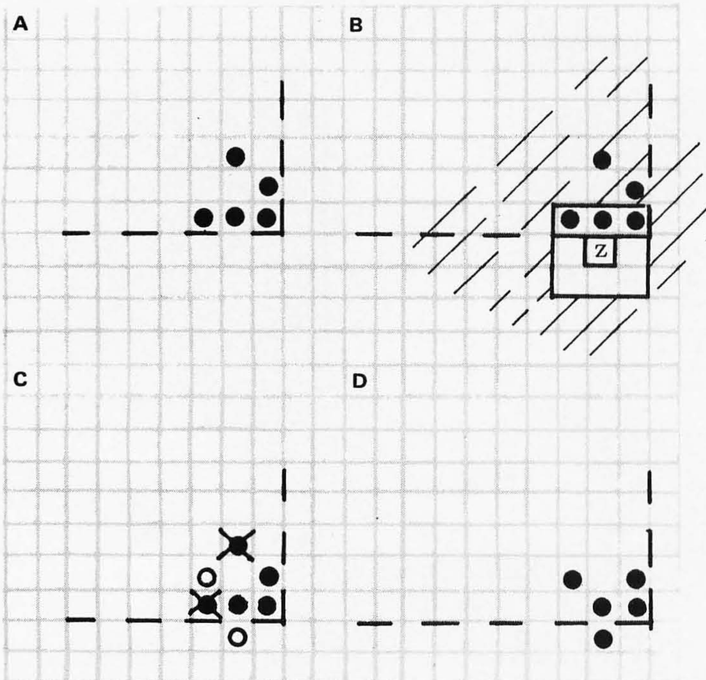
The LIFE game treats each location of the grid (its "squares") as a place where a "cell" might live. If the place is empty, a logical "0" value will be used in the computer memory; if the place is occupied, the "cell" will be indicated by a logical "1" value. The rules of the LIFE algorithm are defined in terms of this idea of a "cell" (logic 1) or "no cell" (logic 0) at every point in the universe of the grid. Fig. 1(a) illustrates a single live cell on a section of graph paper as I might record it when I work out the LIFE process by hand. Fig. 1(b) shows a similar section of the computer memory in which bits ("0" mostly, but "1" for the cell) stand for the content or lack of content of a square on the grid. Fig. 1(c) shows a third view – the output of a program which puts the computer memory bits of the grid onto a graphics display.

Look again at Fig. 1(a). The "cell" on the graph paper grid is a black dot placed in some location. Count the number of graph paper squares which directly surround the live "cell" location. There are 8 possible places which are "nearest neighbors" to the place held by the live cell. Similarly, if you pick an arbitrary square on the graph paper, you can count up its nearest neighbors and find 8 of them also. The rules of the LIFE algorithm concern how to determine whether to place a "cell" in a particular square of the grid for the "next generation", given the present content of that square and its 8 nearest neighbors.

What are the properties of a specific grid location of the game? I've already mentioned its binary valued nature (it has a "cell" or it doesn't) and its neighbors. One more property which is crucial to the game of LIFE is that of the "state" of its 8 nearest neighbor squares. For LIFE, the "state" of the neighbors of a grid location is defined as "the number of occupied neighbors." In the examples of Fig. 1, the "state" of the grid location with the live cell is thus "0" (no neighboring cells), and the state of any cell location which touches the single live cell's location is "1". If I were to fill the entire graph paper or its memory equivalent with live cells, the state of any grid location in the middle would be "8".

Stated in words, the rules of the LIFE algorithm determine the content of each grid location in the "next generation" in terms of its present content and the state of its nearest neighbor grid locations. The rules divide into two groups depending upon the present content of the grid location whose "next generation" value is to be calculated:

Fig. 2. (a) A "glider" generation #n. (b) Examining location "Z" and its nearest neighbors. (c) What has to change for generation #n+1. (d) The second phase of the glider (generation #n+1).



Rule 1. *LIVE CELL LOCATIONS. If* the location to be evolved has a live "cell" at present ("this generation") then,

1.1 *Starving for Affection. If* the location to be evolved has a state of 0 or 1, there will be no cell at the location in the next generation. Metaphorically, if the cell has only one or no nearest neighbors it will die out for lack of interaction with other members of its species.

1.2 *Status Quo. If* the location to be evolved has a state of 2 or 3, the present live cell will live into the tomorrow of the next generation.

1.3 *Overpopulation. If* the location to be evolved has a state of 4 thru 8, there will be no cell at the location in the next generation. Metaphorically, the cell has been crowded out by overpopulation on a local basis.

Rule 2. *EMPTY LOCATIONS. If* the location to be evolved has no live "cell" at present ("this generation") then,

2.1 *The Sex Life of Cells. If* the location to be evolved has a state of 3, a new cell will be "born" in the formerly empty location for the "next generation." Metaphorically, the three neighboring "parent" cells have decided it is time to have a child.

2.2 *Emptiness. If* the location to be evolved does not have three cells in neighboring locations, it will remain empty.

This is the simplest set of rules for the LIFE algorithm, a version which will allow you to begin experimenting with patterns and the evolution of patterns. More complicated extensions can be made to provide an actual interactive (two people) competitive game version; an interesting variation I once implemented is a LIFE game with "genetics." In the genetics variation, each grid location (graph paper square) is represented in the computer as a "character" – an 8 bit byte – of memory. The character in the square is the "gene" pattern of that cell. Then, when rule 2.1 is implemented, LIFE with genetics uses a set of genetic evolution rules to determine which character will be put in the newborn cell based upon the "genes" of the parents. (This genetic evolution program for LIFE was written for my associates at Intermetrics, Inc., as a test program to try out a new compiler's output.)

How Do You Use The Facts of Life?

To illustrate the facts of LIFE, a hand-worked example is a valuable tool of understanding. Consider a "typical" pattern of LIFE as shown in Fig. 2(a). Fig. 2(a) shows what LIFE addicts call a "glider" for reasons which will become clear a little bit later in this article. The glider pattern of Fig. 2(a) consists of the five cells indicated by black dots, and their positions relative to one another. I have also indicated a dotted line in all the illustrations of Figs. 2 and 3 as a fixed reference point in the grid.

The algorithm for evolving one generation to the next is illustrated for one grid location in Fig. 2(b). The LIFE program will examine each location in the grid one by one. This examination is used to figure out what the content of the cell will be in the next generation according to the facts of LIFE. Since these facts only require knowledge of the given grid location Z and its 8 nearest neighbor locations, Fig. 2(b) depicts a box of 9 squares including Z. The rest of the universe is shown shaded. To determine what grid-space location Z will be like in the next generation, the LIFE program first counts up the live cells in all the nearest-neighbor positions. The count is the "state" of Z. In this case there are 3 live cells on the top edge of the box containing Z. Then, the program chooses which rule to use depending upon whether or not location Z has a cell. In this case, Z is empty so the "empty location" set of rules (numbers 2.1 or 2.2) is used. Since the state of Z is 3, rule 2.1 applies and a cell will be born in location Z for the next generation.

Now if I had a true "cellular automaton" to implement the LIFE program, all grid locations would be evolved "simultaneously" – and very quickly – in the computation of the next generation. In point of fact, however, I have

a computer which can only handle 8 (or 16) bits at a time which are stored in words of memory. For small microcomputers, these bits for the LIFE grid will be stored as "packed" bit strings and will be accessed by a series of subroutines which will be described in LIFE Line when the time comes. I have to sequentially look at every bit of the internal LIFE grid of the program and examine its *old* nearest neighbors in order to calculate its new value. I emphasize *old* for the following reason: if I store the new value of the grid location just evolved back into that location with no provision to recall its old value, I'll end up with a mixture of old and new data when I look at the next grid location in the row. That mixture is not part of the rules and constitutes a "faulty" program for evolution. It turns out to be sufficient to remember all the data in one previous row before it was changed in order to calculate the next row after the change. Similar problems of keeping track of partially updated data often occur in computer programming, to be solved by the identical technique of temporarily remembering a copy of the un-updated data.

In Fig. 2(c), the result of examining all the grid locations in the vicinity of the glider of Fig. 2(a) is illustrated. The changes are indicated by three notations for cells:

○ – this indicates a new cell generated by rule 2.1

⊗ – this indicates an old cell which dies by rules 1.1 or 1.3

● – this indicates an old cell which is retained by rule 1.2

Generation "n+1" of the grid of LIFE is illustrated in Fig. 2(d), which was obtained by "executing" the changes noted in Fig. 2(c). When the LIFE program is run, all this is done automatically for each point in the grid – resulting in a new generation as soon as the computer can complete all the calculations. The patterns will be seen to "evolve" in real time as new generations are calculated and sent to the scope output. One "dot" on the scope display corresponds to each live cell of the grid pattern. Fig. 3, (a), (b) and (c), continue the pattern evolution illustrated in Fig. 2 for the "glider". In Fig. 3(a), changes to generation n+1 are indicated with the same notation as was used in Fig. 2(c). The resulting generation n+2 pattern is shown at the right. Fig. 3(b) shows the changes from generation n+2 to generation n+3, and 3(c) shows the change going to generation n+4.

One of the most interesting features of the LIFE game is the evolution of patterns which "move" across a graphics display device. With a fast enough processor, a glider such as the one used in this example will "glide" to the lower right of the screen at a breakneck speed, going off into limbo at the edge – or if the program is sufficiently "smart", reappearing elsewhere on the screen due to a "wrap-around". The reason that the glider gets its name is because of its motion attributes. Note now the fourth generation ("n+4") in the sequence repeats the original glider pattern, *but* has moved one unit along a diagonal of the LIFE grid toward the lower right. (The reference line shows this movement.) It took four generations for the glider pattern to regenerate its original form, which defines the "period" of this pattern. When you get your graphics interface up and running, you will find numerous other classes of patterns, some of which have periods which run into hundreds of generations. There are also other forms of moving patterns similar to the glider.

What Do I Need to Implement LIFE?

The fun part of LIFE is to experiment with patterns of cells and observe how the evolution from generation to generation changes with patterns and classes of patterns. In the lexicon of LIFE lovers, there are whole classes of "gliders", "space ships", "blocks", the "blinkers", "beehives", the "PI" and other patterns. You'll be able to set up initial configurations of these and other patterns, and observe the course of evolution using the hardware/software system

Fig. 3. (a) Third phase of the glider. (b) Fourth phase of the glider. (c) Back to the first phase, but displaced!

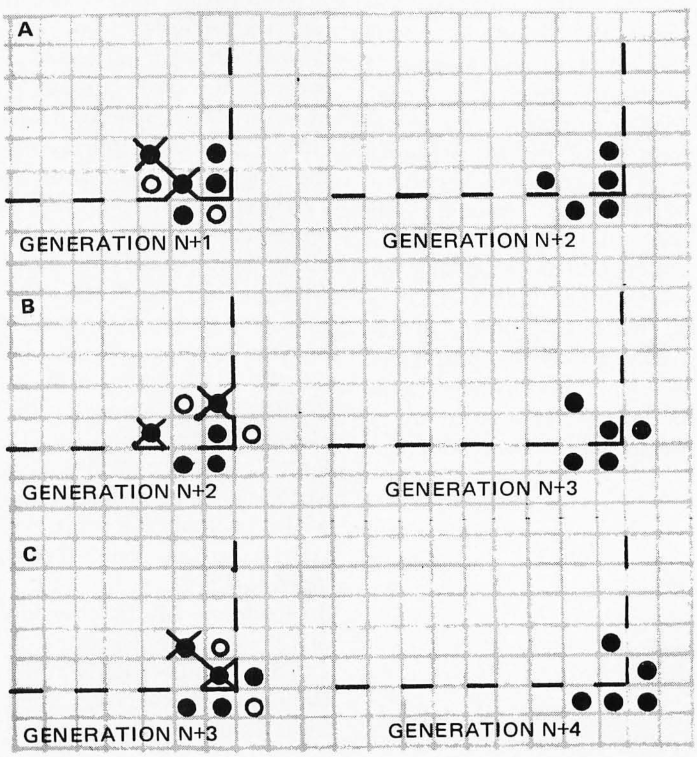
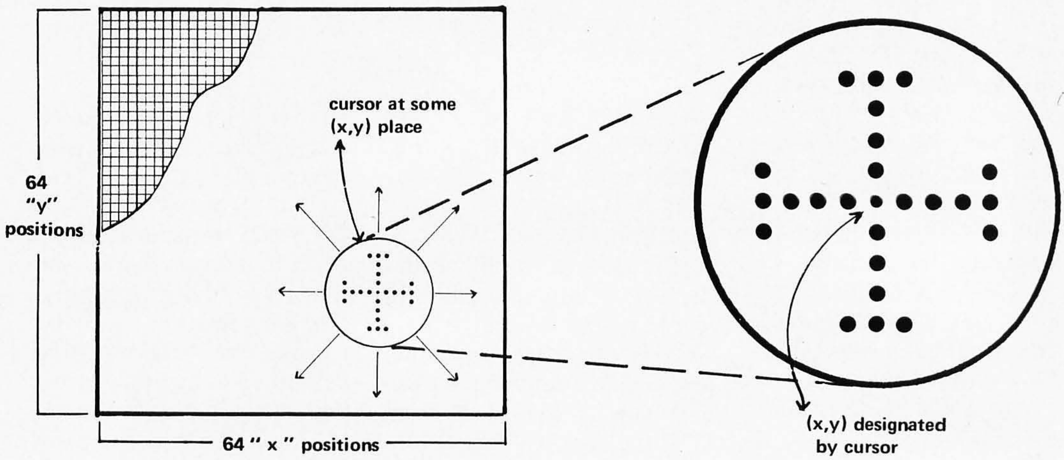


Fig. 4. The LIFE grid display with cursor detail (showing suggested pattern).



concepts of LIFE Line. The hardware requirements of this application's first simple form are three:

1. *An input method.* The best all around input you can get for your computer is an ASCII encoded typewriter keyboard. This hardware will be assumed, with 7-bit ASCII codes used in the examples of programs. If you feel like embellishing *the program* with special hardware, a "paddle" with several keys can be wired in parallel with your main keyboard to control the special functions of the LIFE program. The input keys used to control the display will require a keyboard which can detect two simultaneous (or three) keys being pressed. A normal ASCII encoded keyboard with an LSI encoding chip will not work "as is" in this application since pressing two keys (other than *control* or *shift* and one other) will be resolved into two characters. An alternate "paddle" type of arrangement is to use a single input port with one switch key switch for each bit of the port, debounced by software. A keyboard which is encoded by a diode matrix can be used since the diode matrix will give a new code (logical sum) based upon which keys were depressed.

2. *A processor.* The game can be implemented on any conventional computer. As a measure of capacity, however, the simple form will assume a 64x64 bit array for the playing field, and an available home brew processor such as an Intel 8080 (i.e.: Altair), Motorola 6800, or National PACE. The total programming capacity of your memory should be roughly 4000 8-bit words, or 2000 16-bit words; the playing field will require 512 8-bit words, or 256 16-bit words – and programming will include a set of subroutines to access individual bits.

3. *A display.* My first version of LIFE was implemented on a PDP-6 in FORTRAN at the University of Rochester when I was a student. That program used a direct link out to a DEC Scope controlled by a PDP-8 – with a teletype for input. I have since implemented life programs using character-oriented terminal output and line printers.

The display to be used for LIFE Line purposes I'll leave undefined in detail, but with the following characteristics: It should have an X-Y selection of coordinates for display elements (LIFE grid locations), which can be individually controlled. Its size will be assumed 64x64.

A Note Regarding Speed

The LIFE algorithm to be illustrated in LIFE Line is optimized fairly well for speed – a requirement which will become obvious in the context of your own system if you use a typical microprocessor. With a fairly large pattern of cells, it may take as much as a minute or more to compute the next generation. Trading off against speed is memory size – use of a packed bit structure is necessary if the matrix and programs are to fit in a micro computer which is inexpensive. But the packed bit structure requires time to access bits (eg: the shift/rotate instructions several times might be used in the access process). I predict that the program will be "dreadfully slow" if run on an 8008, and perhaps passably quick if you use a 6800 or 8080. ("Passably quick" means under 10 seconds per generation.) A used third-generation mini (high speed TTL) would be ideal.

User Features

No application is complete without taking into consideration the *user* of the system. The interface which controls the system is an important section of the design. There is a temptation on the part of individuals such as you or I to say words to the effect: "Since I am making it for me, who the heck cares about the user interface." But! Removing the system from the working product realm to the purely personal realm does not eliminate the need to design a

usable system. You have at least one user to think of — yourself! In point of fact, however, I doubt that any reader who builds a scope or TV graphics interface will be able to resist the temptation to show it off to his or her family and friends; so, even for "fun" systems, consideration of users is still a major input to the design.

Birth — the cursor leaves a path of "cells," illuminated points.

Death — cells in the cursor's path are eliminated.

The user interface for the LIFE program will provide the following functions to enable a pattern to be drawn on the screen and initiated:

1. *Cursor.* The display output should provide a "cursor" which is maintained all the time by a subroutine in the software at a given "X" and "Y" position of the matrix. Fig. 4 illustrates the point matrix of the screen (here assumed 64x64) and the cursor pattern. The cursor is a visual feedback through the display to the user of the LIFE program, illustrating where the program will place or erase information. Fig. 4 shows a blow-up of one possible cursor pattern.

Two additional features are required for a useful cursor output of the program for LIFE. These are:

— A blinking feature. Suppose you have filled the screen with a complicated pattern drawn with the cursor controls described below. A significant number of the screen points are now filled with dots — and there will be a strong tendency to confuse the cursor pattern of Fig. 4 with the actual data pattern you have entered. A "blink" feature can be built into the programs which create the cursor so that you will always be able to distinguish it by its flashes.

— A blanking feature. For the LIFE game, a necessary attribute of cursor control is the ability to blank out the cursor during the actual evolution of patterns. I consider this necessary due to observation of a demonstration LIFE program for one desk top programmable CRT terminal: its cursor is always present and mildly annoying when the LIFE game is in operation.

A basic way to make the cursor disappear from view at certain times is to require active control by cursor display routines when the program is in its input mode. If the LIFE program leaves the input mode to go evolve some patterns, the cursor will die a natural death until the active maintenance is resumed on return to the input mode.

2. *Cursor Control.* The whole purpose of the cursor is to provide a means of feeding back to you — the user — the current grid location the LIFE program is pondering. Movement of the cursor provides the opportunity for three types of data entry to the program:

— Positioning of the Cursor. By simply moving the cursor under control of the keyboard (see below) you can direct the LIFE program's attention to different parts of the screen.

— Sowing Seeds of LIFE. By moving the cursor while indicating a "birth" function, the cursor will leave a trail of "cells" indicated in the display by illuminated points. (One keyboard key is required for this function.)

— The Grim Reaper. By moving the cursor while indicating a "death" function, any cells in the path of the cursor will be eliminated, by turning off the corresponding display point. (One keyboard key is required for this function.)

Motion control is also used to enter data. By picking a data key and at the same time depressing one or two of the cursor direction keys, a "trail" will be left. A timing loop in the input program will be used to set a reasonable motion rate in the X (horizontal) and Y (vertical) directions, so that the data entry will be performed automatically as long as the keys are depressed. The motion control keys and useful combinations are illustrated in Fig. 5.

3. *Program Control Commands.* This is the section of the LIFE program design which is the software analog of the "backplane" data bus concept in a hardware system. LIFE Line concerns a modular LIFE program which will be subject to many variations and improvements.

KILLING TWO BIRDS WITH ONE STONE, or "HOW I DESIGNED A GENERAL INTERACTIVE GRAPHICS SOFTWARE INITIALIZATION PACKAGE IN THE GUISE OF A SPECIFIC APPLICATION.

The ideas contained in this article are by no means limited to control of the graphics display type of device in the LIFE context used for this application. The only necessary connection between the LIFE program proper and the display "drawing" and updating functions is in the existence of several subroutines needed to turn on/turn off selected points, and the ability of the display input ("drawing") routines to call the LIFE program. One logical extension of the program control mechanisms to be included in LIFE Line is to allow the invocation (ie: activation, calling, etc.) of other programs and games which use the display.

When the "drawing" routines are up and running, even before you hook up the LIFE algorithm proper, you'll be able to manipulate the contents of the scope under software control and draw pictures on the screen.

Fig. 5. Cursor motion control commands.

The following commands (one key on your keyboard for each) are used to simply move the cursor in one of the grid directions at a rate set by the cursor control software:

Typical "Key Tops"

(UP)	or	(↑)	Move toward top of screen.
(DOWN)	or	(↓)	Move toward bottom of screen.
(LEFT)	or	(←)	Move left on the screen.
(RIGHT)	or	(→)	Move right on the screen.

The following combinations can be used to achieve motion in diagonal directions:

Toward Upper Right Corner – PRESS	(UP)	AND	(RIGHT)	AT THE SAME TIME.
Toward Lower Right Corner – PRESS	(DOWN)	AND	(RIGHT)	AT THE SAME TIME.
Toward Lower Left Corner – PRESS	(DOWN)	AND	(LEFT)	AT THE SAME TIME.
Toward Upper Left Corner – PRESS	(UP)	AND	(LEFT)	AT THE SAME TIME.

Remember that all eight of these possibilities can be used to "sow the seeds" or erase data if the appropriate data key is pressed simultaneously.

The first demonstration of LIFE in these pages is just the bare bones of a LIFE program. When it is fully described you will see the input display routines, the evolution algorithm, the program control *mechanism* and little else. The program control *mechanism*, however, is quite general and will be used to integrate additional commands, variations on LIFE, etc. The means of achieving this modularity is a set of "hooks" which enable you to add commands beyond the bare minimum by coordinating new modules with the program. The following is a minimum set of program control commands for the first version:

RUN – a key assigned to this function will terminate the input ("drawing") mode, and begin the "run mode."

DRAW – a key assigned to this function will be tested during the "run" mode to cause a return to the "draw" mode.

CLEAR – a key assigned to this function will be used to clear the screen in the "drawing" mode, leaving only the cursor and a blank screen.

The above features are only a minimum set of user controls for LIFE. Additional program control commands which will prove invaluable when added include:

SAVE/RESTORE – commands to write and read LIFE patterns on cassette tape or other mass storage device in your home brew system.

INITIALIZATION – functional key entries for the generation of various "standard" LIFE patterns placed at the current cursor location.

Next month, LIFE Line will enter into the realm of software design to describe the LIFE program software in more detail.

LIFE Line Glossary.

Communication of meaning requires definition of terms. The following is a listing of selected terms used in LIFE Line with short explanations. The terms which are marked "L" are primarily significant only in the LIFE application – all others are fairly general terms.

"Active Control" – in the LIFE example, a desired requirement for the cursor is that it disappear automatically if not continually refreshed. This can be accomplished in software by instituting a "garbage sweeper" for the screen which clears the screen memory periodically and updates from the latest non-cursor sources of data. Normally, the cursor control/display subroutine would be called after the screen is updated – but if the cursor control routine is not called, the cursor will be absent after garbage sweeping. The cursor is thus said to require "active control" because it must be explicitly posted on the screen following the garbage sweeping operation if it is to appear at all. (L)

"Algorithm" – this term has a formal mathematical origin as the generalized methodology for arriving at some result. In the computer science area, it retains this definition: an algorithm is the most general processing required to achieve some result. "Algorithm" is a term which includes the term "program" in the following sense: a program is an algorithm (general) as written and coded for a specific system.

"Application" – an application is a specific system designed to accomplish some goal. In the computer systems area, applications are generally composed of hardware and software components which must "play together" to accomplish the desired functions. The LIFE Line's target – a working game of LIFE – is an example of an application.

"Backplane Bus" – the hardware concept of a set of wired connections between identical terminals of multiple sockets. In modular systems, the common wiring makes each socket identical to every other socket. Hardware modules can then be inserted without regard to position in the cabinet containing the equipment.

"Cellular Automata" – conventional computers employ a serial or sequential method of processing. One instruction, then the next, is executed in a time-ordered sequence. The "cellular automata" concept is one way of visualizing large and complicated parallel computing elements. Hypothetically, the LIFE game could be played by such a cellular computer, one which calculates each matrix element simultaneously. In the present state of computer technology, this is not possible, so you have to settle for a simulation of the parallel computation's result, using a serially executing program.(L)

"Coding" – the process of translating a functional specification of a program or routine into a set of machine readable elements for actual use in a computer. Coding can mean writing FORTRAN statements, writing PL/1 statements, writing assembly language statements, or . . . if you have no compiler, coding is the writing of machine codes directly onto a sheet of paper using tables of op codes, an eraser and patience.

"Cursor" – a mark on a display screen used to identify a particular place. This interpretation is an electronic adaptation of the standard definition in Webster.

"Evolution" – patterns in the game of LIFE change from generation to generation according to the rules. The sequence of such changes can loosely be called the evolution of the pattern.(L)

"Feedback" – in the context of system development, feedback is the use of observed system behavior to modify and improve the design of the system.

"Functional Specification" – a functional specification of a system is one which describes "what" the system must do, more or less independent of any technology which is required to make the "what" work. It is easy to come up with loose functional specifications – the hard part is to refine the specification and pin it down to something which is "do-able" in a given context of technology. I have a functional specification in my mind, for instance, of a useful interplanetary travel method – but whether or not I ever see such a system depends upon advances in physics, engineering and economic understanding. *BYTE* often concerns itself with functional specifications of much more "do-able" systems which readers can and will implement on home computers.

"Generation" – this term in the LIFE context means the present "state" of all the locations in the "universe of the grid" at some point in time.(L)

"Implement" – technical jargon verb for the creation of a system or element of a system. A hardware designer might implement a controller or a CPU; a software programmer implements a system of programs; a systems designer implements a hardware/software combination which achieves a desired functional end.

"Indexing" – the technique of referencing data in collection of similar items by means of numerical "indices." In the LIFE Line example, the collection is that of the 64x64 array of bits in the computer representation of "grid space." Indexing by row and by column is used to pick a particular bit within this array when the program requires the data.

"Interact" – when a system "interacts" with "something/person" it is operating under an algorithm which allows conditional behavior dependent upon data. The data is obtained from the "something/person" and may in fact be influenced by previous interactions as well as new inputs. In many computer contexts "interact" has the additional implication of "quick" response in "real time." Thus when you think of an "interactive" terminal or game, you think of a computer programmed so that it keeps up with the inputs from the human operator.

"Lexicon" – the list of buzzwords in any given field. This glossary is a subset of a lexicon coupled with explanations. In compiler and language design, "lexical analysis" is a derivative of this term concerned with language keywords and their relation to a grammar.

"n", "n+1", "n+2"... – when it is useful to specify a sequence of things, where no particular number is intended, a "relative" notation of the sequence is useful. "n" is some arbitrary number; "n+1" is one number greater than an arbitrary number, and so on. When I say "generation n+1" of LIFE, I mean the next generation after generation "n" where "n" is arbitrary.

A suitable LIFE display peripheral is an oscilloscope graphics interface such as the Digital Graphic Display Oscilloscope Interface designed by James Hogenson and printed in the May 1975 issue of ECS Magazine, *the predecessor to* BYTE. *The graphics interface article will be expanded and published in* BYTE *No. 2, October 1975. Until supplies are exhausted, back issues of May ECS (and earlier articles) can be ordered at $2 each. Orders and inquiries regarding ECS back issues should be sent to M. P. Publishing, Box 378, Belmont MA 02178.*

"Partitioning" – the technique of "divide and conquer." Rather than view a complicated system as a monolithic blob of "function," an extremely useful design method is to partition the system into little "bloblets" of function which are easy to understand. Hardware designers of CPUs thus think of MSI chips as sub-elements in partitioning; hardware systems designers think of CPUs and peripherals and memories as sub-elements of partitioning, and software designers consider divisions of complicated programs and program libraries as their sub-elements.

"State" – the present condition of some system, or elements of the system. This term applies to any system which has "memory" to distinguish one possible "state" from another. The term applies equally well to small sub-elements of a system such as the bits of a memory: in the LIFE Line context, the "state" of a single grid location is a number from 0 to 8 counting how many "neighbor cells" are present.

"System" – the most general of all general purpose terms. A system is a collection of component elements (technological, hardware, software, human-interface) selected to play together according to some design or purpose. A system is a human-invented way of doing things.

"Undefined in Detail" – I know what is needed, can specify its interface, but am not at present supplying the detail design. This is a useful attitude since it allows for "plug compatible" designs differing widely in their internal principles of operation. A similar expression would be to call the subsystem in question (the graphic display mentioned in this LIFE Line example) a "black box" and leave it at that. (Software always seems to reference hardware in this way, and hardware does the same for software.) A synonym for the attitude is the mathematician's way of saying "in principle there exists a solution!" without telling you what it is.

"Universe of the Grid" – this is the set of all possible places in which a LIFE cell could be placed. These places are called "grid locations".(L)

MITS Altair Computer Report II

MITS Announces Lower Memory Prices!

On July 1, 1975, **MITS lowered the price** of the Altair 1K Static Memory Card (88-1MCS). The kit price was dropped from $176 to **just $97** while the assembled price was dropped from $209 to $139.

This price reduction was made possible by a reduction in the price of the Altair 1K 8101 memory chips.

Also affected was the price of 88-MM 256 byte (word) memory modules. The $53 kit price was lowered to just $14 and the $61 assembled price to $26.

Altair BASIC—Not Just Anybody's BASIC

Altair BASIC is an easy-to-use programming language that can solve applications problems in business, science and education.

You will find that with only a few hours of using BASIC that you can already write programs with an ease that few other computer languages can match.

Altair BASIC doesn't compromise power for simplicity. While it is one of the simplest computer languages in existence, it is also a very powerful language.

ALTAIR BASIC comes in three versions. The first of these is a 4K BASIC designed to run in an Altair with as little as 4,000 words of memory. This powerful BASIC language has **6 functions** (RND, SQR, SIN, ABS, INT, and SGN) in addition to **15 statements** (IF . . . THEN, GOSUB, RETURN, FOR, NEXT, READ, INPUT, END, DATA GOTO, LET, DIM, REM, RESTORE, PRINT, STOP) and **4 commands** (LIST, RUN, CLEAR, SCRATCH).

The second ALTAIR BASIC option is the 8K BASIC designed to run in an Altair with as little as 8,000 words of memory. This BASIC language is the same as the 4K BASIC only with **8 additional functions** (COS, LOG, EXP, TAN, ATN, INP, FRE, POS) and **4 additional statements** (ON . . . GOTO, ON . . . GOSUB, OUT, DEF) and **1 additional command** (CONT). This BASIC has a multitude of advanced STRING functions and it can be used to control low speed devices—**features not normally found in many BASIC languages.**

The third ALTAIR BASIC is the EXTENDED BASIC version designed to run on an Altair with as little as 12,000 words of memory. It is the same as the 8K BASIC with the addition of PRINT USING, DISK I/O, and double precision (13 digit accuracy) add, substract, multiply and divide.

Altair BASIC is only the beginning. MITS is currently engaged in an extensive software development program. Other software now available includes an Assembler, System Monitor, and Text Editor.

Altair software comes with complete documentation.

One Month Specials

The Altair Users Group is quite possibly the largest computer hobbyist organization in the World. It is both a means of communication among Altair Users and a method of building a comprehensive library of Altair programs. All Altair 8800 owners are entitled to a free, one year membership in this group.

For one month only, you can become an Associate Member for one year at a reduced rate of $10 (regularly $30). Among other benefits you will receive a subscription to the monthly publication, **Computer Notes,** which contains complete update information on Altair hardware and software developments, programming tips, general computer articles and other useful information.

Now available is the **Altair Software Documentation Book I** which contains technical data on the Altair Assembler, Text Editor, System Monitor and BASIC language software. This documentation is free to purchasers of Altair BASIC. For one month only, it is being offered for only $7.50 (regularly $10).

Offers good until September 30, 1975.

The 1K Static Memory Card contains 1024 bytes of memory with a maximum access time of 850 nanoseconds.

Now ready for production is the new Altair 2K Static Memory Card (88-2MCS) with 2048 bytes of memory. Like the 1K Static Memory this new card contains memory protect features and provisions for disabling the ready.

It has a maximum access time of 850 nanoseconds and is engineered with the finest components available. It is inexpensively priced at $145 kit and $195 assembled.

HARDWARE PRICES:

Altair Computer kit with complete assembly instructions	$439
Assembled and tested Altair Computer	$621
1,024 Byte Static Memory Card	$97 kit and $139 assembled
2,048 Byte Static Memory Card	$145 kit and $195 assembled
4,096 Byte Dynamic Memory Card	$264 kit and $338 assembled
Full Parallel Interface Card	$92 kit and $114 assembled
Serial Interface Card RS232)	$119 kit and $138 assembled
Serial Interface Card (TTL or Teletype)	$124 kit and $146 assembled
COMTER II*	$780 kit and $920 assembled

*The Comter II Computer Terminal has a full alpha-numeric keyboard and a highly readable 32-character display. It has its own internal memory of 256 characters and complete cursor control. Also has its own built-in audio cassette interface that allows you to connect the Comter II to any tape recorder for both storing data from the computer and feeding it into the computer. Requires an RS232 Interface Card.

SOFTWARE PRICES:

Altair 4K BASIC	$350
Purchasers of an Altair 8800, 4K of Altair Memory, and Altair Serial I/O or Audio-Cassette I/O	ONLY $60
Altair 8K BASIC	$500
Purchasers of an Altair 8800, 8K of Altair Memory, and Altair Serial I/O or Audio-Cassette I/O	ONLY $75
Altair EXTENDED BASIC	$750
Purchasers of an Altair 8800, 12K of Altair Memory, and Altair Serial I/O or Audio-Cassette I/O	ONLY $150
Altair PACKAGE ONE (assembler, text editor, system monitor)	
Purchasers of an Altair 8800, 8K of Altair Memory, and Altair I/O	ONLY $30

NOTE: When ordering software, specify paper tape or cassette tape.

Warranty: 90 days on parts for kits and 90 days on parts and labor for assembled units. Prices, specifications, and delivery subject to change.

MAIL THIS COUPON TODAY!

☐ Enclosed is check for $________

☐ BankAmericard #________ ☐ or Master Charge #________

☐ Altair 8800 ☐ Kit ☐ Assembled ☐ Options
Include $8 for postage & handling (list on separate sheet)

☐ Altair Users Group Associate ☐ Software Documentation

☐ Please send free literature

NAME________

ADDRESS________

CITY________ STATE & ZIP________

MITS/6328 Linn N.E., Albuquerque, NM 87108 505/265-7553 or 262-1951

MITS

"Creative Electronics"

MITS/6328 Linn N.E., Albuquerque, NM 87108 505/265-7553 or 262-1951

7400N TTL

SN7400N	.16	SN7451N	.27	SN74151N	1.25
SN7401N	.16	SN7453N	.27	SN74153N	1.35
SN7402N	.21	SN7454N	.41	SN74154N	1.25
SN7403N	.16	SN7459A	.25	SN74155N	1.21
SN7404N	.21	SN7460N	.22	SN74156N	1.30
SN7405N	.24	SN7470N	.45	SN74157N	1.30
SN7406N	.45	SN7472N	.39	SN74160N	1.75
SN7407N	.45	SN7473N	.45	SN74161N	1.45
SN7408N	.25	SN7474N	.45	SN74163N	1.65
SN7409N	.25	SN7475N	.80	SN74164N	1.65
SN7410N	.20	SN7476N	.47	SN74165N	1.65
SN7411N	.30	SN7480N	.50	SN74166N	1.70
SN7412N	.42	SN7482N	1.75	SN74167N	5.50
SN7413N	.85	SN7483N	1.15	SN74170N	3.00
SN7414N	.70	SN7485N	1.12	SN74172N	18.00
SN7416N	.43	SN7486N	.45	SN74173N	1.70
SN7417N	.43	SN7488N	3.50	SN74174N	1.95
SN7418N	.25	SN7489N	3.00	SN74175N	1.95
SN7420N	.21	SN7490N	.59	SN74176N	.90
SN7421N	.39	SN7491N	1.20	SN74177N	.90
SN7423N	.37	SN7492N	.82	SN74180N	1.05
SN7425N	.43	SN7493N	.82	SN74181N	3.55
SN7426N	.31	SN7494N	.91	SN74182N	.95
SN7427N	.37	SN7495N	.91	SN74184N	2.30
SN7429N	.42	SN7496N	.91	SN74185N	2.20
SN7430N	.26	SN74100N	1.25	SN74187N	6.00
SN7432N	.31	SN74107N	.49	SN74190N	1.50
SN7437N	.47	SN74121N	.55	SN74191N	1.50
SN7438N	.40	SN74122N	.49	SN74192N	1.50
SN7439A	.25	SN74123N	1.05	SN74193N	1.40
SN7440N	.21	SN74125N	.60	SN74194N	1.45
SN7441N	1.10	SN74126N	.81	SN74195N	1.00
SN7442N	1.08	SN74132N	3.00	SN74196N	1.25
SN7443N	1.05	SN74141N	1.15	SN74197N	1.00
SN7444N	1.10	SN74142N	6.50	SN74198N	2.25
SN7445N	1.10	SN74143N	7.00	SN74199N	2.25
SN7446N	1.15	SN74144N	7.00	SN74200N	7.00
SN7447N	.89	SN74145N	1.15	SN74251N	2.50
SN7448N	.99	SN74148N	2.50	SN74284N	6.00
SN7450N	.26	SN74150N	1.10	SN74285N	6.00

20% Discount for 100 Combined 7400's

CMOS

CD4000	.29			74C10N	.65
CD4001	.29			74C20N	.65
CD4002	.29	CD4030	.65	74C30N	.65
CD4006	2.50	CD4035	1.85	74C42N	2.15
CD4007	.29	CD4040	2.45	74C73N	1.50
CD4009	.59	CD4042	1.90	74C74	1.15
CD4010	.59	CD4044	1.50	74C90N	3.00
CD4011	.29	CD4046	2.51	74C95N	2.00
CD4012	.29	CD4047	2.75	74C107N	1.25
CD4013	.53	CD4049	.79	74C151	2.90
CD4016	.69	CD4050	.79	74C154	3.00
CD4017	1.35	CD4051	2.98	74C157	2.15
CD4019	.69	CD4053	2.98	74C160	3.25
CD4020	1.69	CD4060	3.25	74C161	3.25
CD4022	1.25	CD4066	1.75	74C163	3.00
CD4023	.29	CD4069	.45	74C164	3.25
CD4024	1.50	CD4071	.45	74C173	2.60
CD4025	.34	CD4081	.45	74C193	2.75
CD4027	.85	74C00N	.39	74C195	2.75
CD4028	1.65	74C02N	.55	80C97	1.50
CD4029	2.90	74C04N	.75		

LINEAR

LM100H	15.00			LM1310N	2.95
LM106H	2.50			LM1351N	1.65
LM171H	3.75	LM373N	3.25	LM1414N	1.75
LM212H	7.00	LM377N	4.00	LM1458C	.65
LM300H	.80	LM380N	1.39	LM1496N	.95
LM301H	3/1.00	LM380CN	1.05	LM1556V	1.85
LM301CN	3/1.00	LM381N	1.79	LM2111N	1.95
LM302H	.75	LM382N	1.79	LM2901N	2.95
LM304H	1.00	NE501K	8.00	LM3065N	.69
LM305H	.95	NE510A	6.00	LM3900N	.55
LM307CN	.35	NE531H	3.00	LM3905N	.60
LM308H	1.00	NE536T	6.00	LM5556N	1.85
LM308CN	1.00	NE540L	6.00	MC5558V	1.00
LM309H	1.10	NE550N	.79	LM7525N	.90
LM309K	1.25	NE553	2.50	LM7528N	2.20
LM310CN	1.15	NE555V	.75	LM7534N	2.20
LM311H	.90	NE565H	1.25	LM7535N	1.25
LM311N	.90	NE565N	1.95	8038B	4.95
LM318CN	1.50	NE566CN	1.95	LM75450	.49
LM319N	1.30	NE567H	1.25	75451CN	.39
LM319D	9.00	NE567V	1.95	75452CN	.39
LM320K-5	1.35	LM703CN	.45	75453CN	.39
LM320K-5.2	1.35	LM709H	.29	75454CN	.39
LM320K-12	1.35	LM709N	.29	75491CN	.79
LM320K-15	1.35	LM710N	.79	75492CN	.89
LM323K-5	14.00	LM711N	.39	75494CN	.89
LM324N	1.80	LM723N	.55	RCA LINEAR	
LM339N	1.70	LM723H	.55	CA3013	1.70
LM340K-5	1.95	LM733N	1.00	CA3023	2.15
LM340K-12	1.95	LM739N	1.29	CA3035	2.25
LM340K-15	1.95	LM741CH	3/1.00	CA3039	1.35
LM340K-24	1.95	LM741CN	3/1.00	CA3046	1.15
LM340To-5	1.75	LM741-14N	.39	CA3059	2.46
LM340To-6	1.75	LM747H	.79	CA3060	2.80
LM340To-12	1.75	LM747N	.79	CA3080	.85
LM340To-15	1.75	LM748H	.39	CA3083	1.60
LM340To-24	1.75	LM748N	.39	CA3086	.59
LM350N	1.00	LM1303N	.90	CA3089	3.25
LM351CN	.65	LM1304N	1.19	CA3091	8.25
LM370N	1.15	LM1305N	1.40	CA3123	1.85
LM370H	1.15	LM1307N	.85	CA3600	1.75

MICROPROCESSOR COMPONENTS

8008	CPU	$29.95	8111	1024 RAM	$12.95
8080	CPU	149.95	1702A	2K PROM	29.95
7489	64B RAM	2.95	5203Q	2K PROM	19.95
8599	Tri-State 7489	3.50	8223	PROM	3.00
1101	256B RAM	2.25	2401	2K SR	9.95
2102	1K RAM	4.95	2533	1K SSR	11.85
8101	1024 RAM	11.95	AY-5-1013	UART	7.95
7010	1K NMOS	RAM	Retains data w/o Power		29.95
2107	4K RAM	450NS	Access time-22 pin DIP		19.95

Vector General Purpose Logic CARD Board 14.95
*Very High Noise Immunity * Holds 12 ea. 14 pin DIPS
*44 pin Edge Connection

THE KILOBYTE RAM CARD Per Kit 69.95
*Complete 1Kx8 Memory * High Noise Immunity Components
*Single 5v supply *500NS Access Time &Kit includes sockets, ICS & Board

FREE $12.00 Value With each $75. order of Microprocessor components, get a one year subscription to **BYTE** the magazine for computer PHREAKS FREE FREE FREE

8000 SERIES

8091	.59			8552	2.49
8092	.59	8223	*3.00	8554	2.49
8095	1.39	8230	2.59	8810	.79
8121	.89	8263	7.00	8820	2.00
8123	1.59	8267	4.00	8826	3.00
8130	2.19	8280	.75	8830	2.59
8200	2.59	8281	.85	8831	2.59
8210	3.49	8288	1.15	8836	.49
8214	1.69	8520	1.29	8864	2.00
8220	1.69	8551	1.65	8880	1.35

DIGITAL COUNTER UNIT



*Digit Latches
*On Board Oscillator
4 Each – Man7 Displays
* ÷ 10000 Counter
*Complete with Board .... $29.95 Kit

DIGITAL VOLTMETER KIT

0-2 Volt, Auto Polarity 3½ digits (MAN7)
DVM MOS - LSI Design
Size: 2½" x 2¼" x 1½"



$39.95

TTL Logic Probe Kit



Detects TTL levels, pulses, with man 3 readout
$9.95 per kit

DVM Chip Set Siliconix

LD110 Digital A/D Processor 16.00
LD111 Analog A/D Processor 13.00 $28.00 Set

DM8890N

Complete Horiz./Vert. Divider Chain for T.V. Type, Appl. 1.95 Each

4' POWER SUPPLY CORDS ....

Black



.59¢ ea.

POCKET CALCULATOR KIT



5 function plus constant – addressable memory with individual recall – 8 digit display plus overflow – battery saver – uses standard or rechargeable batteries – all necessary parts in ready to assemble form – instructions included. 3" x 5¼" .... $17.50 each
OPTIONS –
115VAC Transformer .... 4.95 each
6 each "N" Alkaline Batteries ..... 2.50 lot

THUMBWHEEL SWITCHES



THUMBWHEEL SWITCH ONLY
SF – Front Mount SR – Rear Mount

Part No.	Description	Price
SF-12	Single Pole 10 Position	$2.50
SR-12	Decimal	3.00
SF-21	10 Position BCD only	2.50
SR-21		3.00

Snap together – No Hardware

SERIES SF Front Mount Assembly ($11.70)



Note: Picture to the left has the following 4 ea. SF-12, 1 ea. SF-EP, 1 ea. SF-BB, 1 ea. SF-DP and 1 ea. SF-HB

ACCESSORIES

Part No.	Description	Price
SF-EP	End Plate (Pair)	.50
SF-DP	Divider Plate (each)	.40
SF-BB	Blank Body (each)	.40
SF-HB	Half Body (each)	.40

SERIES SR Rear Mount Assembly ($13.70)



Note: Picture to the left has the following 4 ea. SR-12, 1 ea. SR-EP, 1 ea. SR-BB, 1 ea. SR-DP and 1 ea. SR-HB

ACCESSORIES

Part No.	Description	Price
SR-EP	End Plates (pair)	$ .50
SR-DP	Divider Plate (each)	.40
SR-BB	Blank Body (each)	.40
SR-HB	Half Body (each)	.40

Ordering: Order desired switch or switches and add necessary accessories for your particular application.

.394" DIAM. TRIMMER

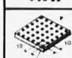


STANDARD RESISTANCE VALUES
MODEL 100Ω 500Ω 1K 2K 5K 10K 20K 100K 200K 1Meg

Model TR 11

RESISTANCES (OHMS)	1–9	10–24
STD100Ω 1 MEGΩ	.35	.30

1/16 VECTOR BOARD



P-Pattern
0.1" Hole Spacing

MATERIAL	STOCK NO.	LONG	WIDE	PRICES 1	20
PHENOLIC	64P44 062XXXP	4.50	6.50	1.72	1.54
	169P44 02XXXP	4.50	17.00	3.69	3.32
EPOXY GLASS	64P44 062	4.50	6.50	2.07	1.86
	84P44 062	4.50	8.50	2.56	2.31
	169P44 062	4.50	17.00	5.04	4.53
	169P84 062	8.50	17.00	9.23	8.26
EPOXY GLASS COPPER CLAD	169P44 062C1	4.50	17.00	6.80	6.12

NEW PROTO BOARD-100



Here's a low cost, big 10 IC capacity breadboard kit with all the quality of QT Sockets and the best of the Proto-Board series . . . complete down to the last nut, bolt and screw. Includes 2 QT-35S Sockets; 1 QT-35B Bus Strip; 2 5-way binding posts; 4 rubber feet; screws, nuts, bolts; and easy assembly instructions.

COMPLETE KIT... $19.95

WALL or T.V. DIGITAL CLOCK



12 or 24 Hour
25' VIEWING DISTANCE
Walnut Case-6" x 3" x 1"
Hr. & Min.-6" High
Seconds-3" High
KIT - All Comp. & Case $39.95
Wired & Assembled 115 Vac $44.95

DISPLAY LEDS

CA	Common Anode	CC	Common Cathode
MAN 1	Com. Ano.	.270	$1.95
MAN 2	5x7 Matrix	.300	4.25
MAN 3	Com. Cath.	.125	.95
MAN 4	Com. Cath.	.187	1.95
MAN 7	Com. Ano.	.30	1.50
DL33	Com. Cath.	.125	1.95
DL747	Com. Ano.	.625	2.50

DISCRETE LEDS



MV 10	Red	5/$1.00
MV 50	Red	6/$1.00
MV 5024	Red	5/$1.00
MV 5024	Green	4/$1.00
MV 5024	Yellow	4/$1.00
MV 5024	Orange	4/$1.00

IC SOLDERTAIL – LOW PROFILE (TIN) SOCKETS

	1-24	25-49	50-100		1-24	25-49	50-100
8 pin	$.19	.18	.17	24 pin	$ .68	.62	.56
14 pin	.23	.22	.21	28 pin	.89	.81	.73
16 pin	.27	.26	.24	36 pin	1.10	.99	.90
18 pin	.46	.41	.37	40 pin	1.25	1.13	.93
22 pin	.65	.59	.53				
SOLDERTAIL STANDARD (TIN)							
14 pin	$.30	.28	.26	28 pin	$ .99	.90	.81
16 pin	.33	.31	.29	36 pin	1.39	1.26	1.15
18 pin	.42	.39	.37	40 pin	1.59	1.45	1.30
24 pin	.59	.54	.49				
SOLDERTAIL STANDARD (GOLD)							
8 pin	$.30	.27	.24	24 pin	$ .70	.63	.57
14 pin	.35	.32	.29	28 pin	1.10	1.00	.90
16 pin	.38	.35	.32	36 pin	1.55	1.40	1.26
18 pin	.52	.47	.43	40 pin	1.75	1.59	1.45
WIRE WRAP SOCKETS (GOLD) LEVEL #3							
10 pin	$.45	.41	.37	24 pin	$1.05	.95	.85
14 pin	.46	.42	.38	28 pin	1.40	1.25	1.10
16 pin	.55	.50	.45	36 pin	1.59	1.45	1.30
18 pin	.75	.68	.62	40 pin	1.75	1.55	1.40

50 PCS. RESISTOR ASSORTMENTS $1.75 PER ASST.

ASST. 1	5 ea:	10 OHM 27 OHM	12 OHM 33 OHM	15 OHM 39 OHM	18 OHM 47 OHM	22 OHM 56 OHM	1/4 WATT 5% = 50 PCS.
ASST. 2	5 ea:	68 OHM 180 OHM	82 OHM 220 OHM	100 OHM 270 OHM	120 OHM 330 OHM	150 OHM 390 OHM	1/4 WATT 5% = 50 PCS.
ASST. 3	5 ea:	470 OHM 1.2K	560 OHM 1.5K	680 OHM 1.8K	820 OHM 2.2K	1K 2.7K	1/4 WATT 5% = 50 PCS.
ASST. 4	5 ea:	3.3K 8.2K	3.9K 10K	4.7K 12K	5.6K 15K	6.8K 18K	1/4 WATT 5% = 50 PCS.
ASST. 5	5 ea:	22K 56K	27K 68K	33K 82K	39K 100K	47K 120K	1/4 WATT 5% = 50 PCS.
ASST. 6	5 ea:	150K 390K	180K 470K	220K 560K	270K 680K	330K 820K	1/4 WATT 5% = 50 PCS.
ASST. 7	5 ea:	1M 2.7M	1.2M 3.3M	1.5M 3.9M	1.8M 4.7M	2.2M 5.6M	1/4 WATT 5% = 50 PCS.

PRIME INTEGRATED CIRCUIT ASSORTMENTS

ASST. 8	3 ea.	SN7400 SN7410	7401 7430	7402 7438	7403 7440	7404 7472	SSI/TTL	$5.95 ASST.
ASST. 9	2 ea.	SN7447 SN74145	7490 74175	7491 74180	74100 74191	74121 74193	MSI/TTL	$9.95 ASST.
ASST. 10	2 ea.	CD4001 CD4016	4002 4017	4011 4019	4012 4023	4013 4030	CMOS	$9.95 ASST.
ASST. 11	2 ea.	LM301T LM311T	301N 565T	302T 567T	307T 741T	309K 741N	LINEAR	$10.95 ASST.

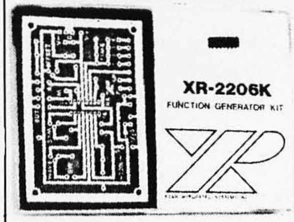
Satisfaction Guaranteed. $5.00 Min. Order. U.S. Funds.
California Residents — Add 6% Sales Tax
Write for FREE 1975S Catalog — Data Sheets .25¢ each

JAMES

P.O. BOX 822, BELMONT, CA. 94002
PHONE ORDERS — (415) 592-8097

IC'S EXAR KITS

FUNCTION GENERATOR KIT



features sine, triangle and square wave; THD 0.5% typ.; AM/FM capability

XR-2206KA **$19.95**
Includes monolithic function generator IC, PC board, and assembly instruction manual.

XR-2206KB **$29.95**
Same as XR-2206KA above and includes external components for PC board.

TIMERS		
XR-555CP	Monolithic Timer	$ 1.10
XR-320P	Precision Timer	1.55
XR-556CP	Dual-555 Timer	1.85
XR-2556CP	Dual Timing Circuit	3.20
XR-2240CP	Programmable Counter/Timer	4.80
PHASE LOCKED LOOPS		
XR-210	FSK Demodulator	5.20
XR-215	High Frequency PLL	6.60
XR-567CP	Tone Decoder (mini DIP)	1.95
XR-567CT	Tone Decoder (TO-5)	1.70
STEREO DECODERS		
XR-1310P	PLL Stereo Decoder	3.20
XR-1310EP	PLL Stereo Decoder	3.20
XR-1800P	PLL Stereo Decoder	3.20
WAVEFORM GENERATORS		
XR-205	Waveform Generator	8.40
XR-2206CP	Monolithic Function Generator	5.50
XR-2207CP	Voltage-Controlled Oscillator	3.85
OTHER EXAR IC'S		
XR-1468CN	Dual + 15V Tracking Regulator	3.85
XR-1488N	Quad Line Driver	5.80
XR-1489AN	Quad Line Receiver	4.80
XR-2208CP	Operational Multiplier	5.20
XR-2211 CP	FSK Demodulator/Tone Decoder	6.70
XR-2261	Monolithic Proportional Servo IC System w/4 ea. Driver Transistor	3.79

Special Requested Items

RC4194	Dual Track V Reg	$5.95	N8T97	$3.00	2533	$11.85
RC4195	* 15v Track Reg	3.25	4024P	2.25	8263	5.95
MC1741	High Slew Op Amp	4.00	2513	11.00	8267	2.75
MC4044P		4.50	2518	7.00	8288	1.15
CA3130	Super CMOS Op Amp	1.49	2524	3.50	8826	3.00
40410	3w PNP	1.75	2525	6.00	8880	1.35
40673	3w NPN	1.75	2527	5.00	7497	5.00

DIODES

(Zener) TYPE	VOLTS	W	PRICE	(Rectifier) TYPE	VOLTS	W	PRICE
IN746	3.3	400m	4/1.00	IN4003	200 PIV	1 AMP	.10
IN751A	5.1	400m	4/1.00	IN4004	400 PIV	1 AMP	.10
IN752	5.6	400m	4/1.00	IN3600	50	200m	6/1.00
IN753	6.2	400m	4/1.00	IN4148	75	10m	15/1.00
IN754	6.8	400m	4/1.00	IN4154	35	10m	12/1.00
IN965B	15	400m	4/1.00	IN4734	5.6	1w	.28
IN5232	5.6	500m	.28	IN4735	6.2	1w	.28
IN5234	6.2	500m	.28	IN4736	6.8	1w	.28
IN5235	6.8	500m	.28	IN4738	8.2	1w	.28
IN5236	7.5	500m	.28	IN4742	12	1w	.28
IN456	25	40m	6/1.00	IN4744	15	1w	.28
IN458	150	7m	6/1.00	IN1183	50 PIV	35 AMP	1.60
IN485A	180	10m	5/1.00	IN1184	100 PIV	35 AMP	1.70
IN4001	50 PIV	1 AMP	.09	IN1186	200 PIV	35 AMP	1.80
IN4002	100 PIV	1 AMP	.10	IN1188	400 PIV	35 AMP	3.00

TRANSISTORS

MPS-A05	5/$1			2N3905	4/$1
2N918	.25	2N2906A	4/$1	2N3906	4/$1
2N2219A	3/$1	2N2907A	5/$1	PN4249	4/$1
2N2221	4/$1	2N3053	2/$1	PN4250	4/$1
2N2222A	5/$1	2N3055	.95	2N4409	5/$1
2N2369	5/$1	2N3725A	2/$1	2N5129	.19
2N2369A	4/$1	2N3903	5/$1	2N5139	.19
2N2484	4/$1	2N3904	4/$1	C106B1-SCR	2/$1

CAPACITOR CORNER

50 VOLT CERAMIC DISC CAPACITORS

	1-9	10-49	50-100		1-9	10-49	50-100
10pf	.05	.04	.03	.001	.05	.04	.035
22 pf	.05	.04	.03	.0047	.05	.04	.035
47 pf	.05	.04	.03	.01	.05	.04	.035
100 pf	.05	.04	.03	.022	.06	.05	.04
220 pf	.05	.04	.03	.047	.06	.05	.04
470 pf	.05	.04	.035	.1	.12	.09	.075

100 VOLT MYLAR FILM CAPACITORS

.001mf	.12	.10	.07	.022mf	.13	.11	.08
.0022	.12	.10	.07	.047mf	.21	.17	.13
.0047mf	.12	.10	.07	.1mf	.27	.23	.17
.01mf	.12	.10	.07	.22mf	.33	.27	.22

± 20% DIPPED TANTALUMS (SOLID) CAPACITORS

.1	35V	.28	.23	.17	1.5	35V	.30	.26	.21
.15	35V	.28	.23	.17	2.2	25V	.31	.27	.22
.22	35V	.28	.23	.17	3.3	25V	.31	.27	.22
.33	35V	.28	.23	.17	4.7	25V	.32	.28	.23
.47	35V	.28	.23	.17	6.8	25V	.36	.31	.25
.68	35V	.28	.23	.17	10	25V	.40	.35	.29
1.0	35V	.28	.23	.17	15	25V	.63	.50	.40

MINIATURE ALUMINUM ELECTROLYTIC CAPACITORS

Axial Lead					Radial Lead				
.47	50	.15	.13	.10	.47	25	.15	.13	.10
1	50	.16	.14	.11	.47	50	.16	-.14	.11
3.3	50	.15	.13	.10	1	16	.15	.13	.10
4.7	25	.16	.14	.12	1	25	.16	.14	.11
10	25	.15	.13	.10	1	50	.16	.14	.11
10	50	.16	.14	.12	4.7	16	.15	.13	.10
22	25	.17	.15	.12	4.7	25	.15	.13	.10
22	50	.24	.20	.18	4.7	50	.16	.14	.11
47	25	.19	.17	.15	10	16	.14	.12	.09
47	50	.25	.21	.19	10	25	.15	.13	.10
100	25	.24	.20	.18	10	50	.16	.14	.12
100	50	.35	.30	.28	47	50	.24	.21	.19
220	25	.32	.28	.25	100	16	.19	.15	.14
220	50	.45	.41	.38	100	25	.24	.20	.18
470	25	.33	.29	.27	100	50	.35	.30	.28
1000	16	.55	.50	.45	220	16	.23	.17	.16
2200	16	.70	.62	.55	470	25	.31	.28	.26

Memory Dumps

Until somebody invents a direct link between human brains, the only way to find out about methods and techniques is to read someone else's "memory dump"... books, magazines and other sources. Associate Editor Dan Fylstra has provided us with three reviews of books which will prove useful in your home brew computer work. These memory dumps are not in hexadecimal or octal – and are definitely "readable."

...CARL



BOOK REVIEW

The Elements of Programming Style *by Brian W. Kernighan and P. J. Plauger. McGraw-Hill, New York, 1974. $3.95.*

This book is *required reading* for anyone who is seriously interested in writing good programs. Even the best programmers (and especially the most clever ones) can profit from reading this book. The authors take *all* their examples of dubious programming practices from textbooks intended to teach programming! Those of us who have learned programming from such textbooks will find many of the points made here useful as well as amusing.

The intent of the book is to teach *programming style*, or the principles of writing well-structured, readable programs that work (in all cases) and are efficient. The approach is pragmatic and down-to-earth, and can be applied to everyday programming problems. All of the examples are in Fortran or PL/I, and can be read and understood by anyone familiar with either language. The elements of style, as the authors point out, are applicable regardless of the language being used, and the principles will be of interest even to assembly language programmers.

There are chapters on writing computational expressions, control structure, input/output and data verification, common blunders, efficiency and instrumentation, and documentation. Each chapter takes a series of example programs, criticizes them, rewrites them with improvements, and then extracts some general principles of good programming practice from the examples. The principles are summarized as a series of short aphorisms which are listed together at the end of the book. Examples are "Don't patch bad code – rewrite it," "Test programs at their boundary values," and "Make sure comments and code agree."

The more programming experience you have, the more you will appreciate this book. Buy a copy for yourself, read it, and keep it around for reference! – d.h.f.

Designing Logic Systems Using State Machines *by Christopher R. Clare. McGraw-Hill, New York, 1973. $9.50.*

This is an advanced text on logic design which will be of interest to anyone embarking on the design of large-scale logic systems. A number of important and valuable ideas are presented here, apparently for the first time. The methods were developed by Tom Osborne at Hewlett-Packard Laboratories and were used in the design of the HP calculators. The main features of the book are the introduction of "Algorithmic State Machines" (ASMs) to describe logic systems, and a comprehensive discussion of logic synthesis using Read-Only Memories (ROMs).

An ASM is something of a cross between a flowchart and a finite state machine (it looks like a flowchart, but has boxes denoting states, with assignments for state

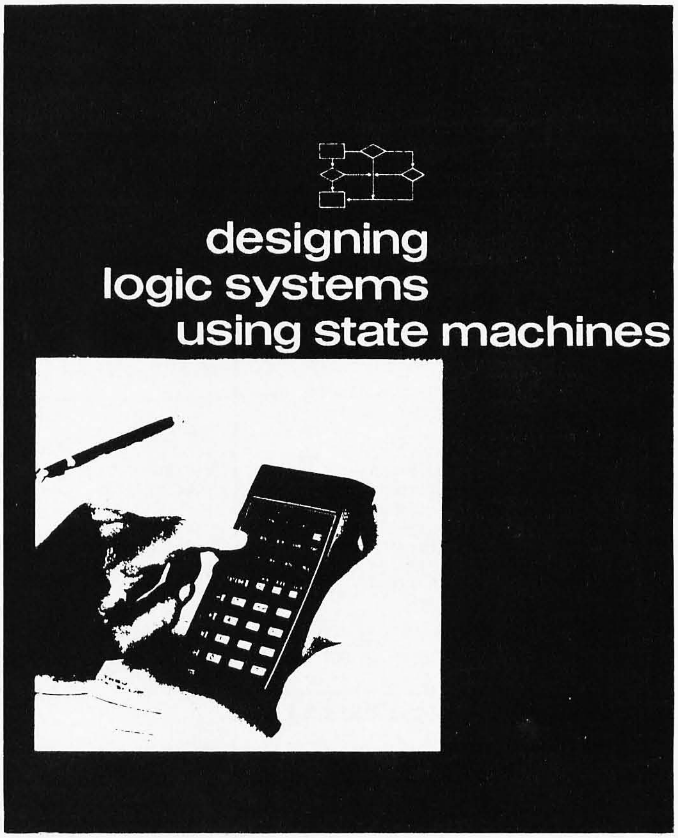
variables). ASMs turn out to be very convenient and intuitive for describing complex logic functions, especially in the initial stages of design.

The chapter on ROM-centered design is probably the most interesting part of the book. It discusses a number of techniques for getting the most out of a ROM, and trading off ROM space and external decoding logic. The material presented here is difficult to find elsewhere, and as the price of LSI chips continues to drop, the use of ROMs is becoming increasingly attractive.

Two other sections of the book are also noteworthy. The introduction discusses the nature of an algorithm – a concept often misunderstood by logic designers – and the value of modularity and functional division. The chapter on "Linked State Machines" introduces the valuable notion of interpretive linking, in which a hierarchy of machines is built up such that each state of a "higher level" machine can be described by the ASM chart of a lower level machine.

Other features of the book are a complete discussion of Karnaugh maps, including techniques for constructing maps for functions of more than four variables, and a brief treatment of logic system simulation and performance evaluation.

This book is suitable only for those with some previous background and experience in logic design. The book is very well organized, but it is tersely written and requires the reader to think and to study the examples. The comments on software linked machines and computer structures, especially those on Turing machines, should not be taken too seriously. The reader who patiently studies this book will profit greatly from the time spent with it.
-d.h.f.



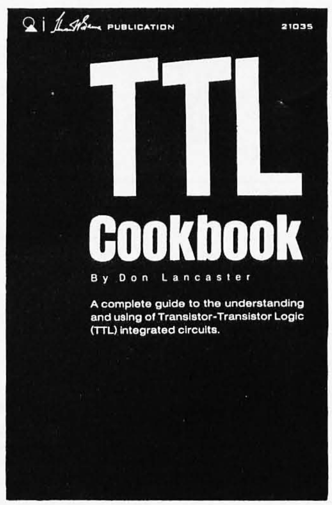
TTL Cookbook *by Don Lancaster. Howard W. Sams & Co., Indianapolis, 1974. $8.95.*

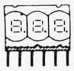
This book should be in the hands of every hobbyist who experiments with digital integrated circuits. It is also recommended for those who prefer to work with "higher-level" microcomputer system elements, since microcomputer applications often require at least a little "random logic" in hardware. The book contains a wealth of practical information, ranging from circuit breadboarding techniques and power supplies to sophisticated design methods using shift registers and binary rate multipliers. Besides providing a good deal of useful information, the book includes many pointers, cautions and "words of wisdom" for the experimenter.

The book consists of eight chapters, one of which provides a list of short descriptions of the most commonly used TTL ICs, and another which outlines a number of interesting projects for the hobbyist.

Many amateurs will find the first chapter, "Some Basics of TTL," especially valuable. It discusses practical matters such as power supply spike decoupling, current requirements, monitoring circuit states, tools, "bad" and "burned-out" ICs, and much more. The chapter on logic is notable for its explanations of positive and negative logic, tri-state and open-collector logic, and data selectors and ROMs. Other chapters cover gates and timer circuits, JK and D-type flipflops, counters, and shift registers and rate multipliers.

The book does *not* discuss traditional design techniques such as Karnaugh maps and state machines. The author argues, with some justification, that these techniques do not often lead to circuits with a minimal number of IC packages and therefore the lowest cost. Instead the book lives up to the promise of its title by providing a tasteful selection of "recipe" circuits -- tried and true ideas -- which the experimenter can put into practice. Perhaps Don Lancaster will follow up this very useful book with another one for "gourmet" experimenters. -d.h.f.





7-Segment Readout 12-PIN DIP

Three digits with right-hand decimal
Plugs into DIP sockets
Similar to (LITRONIX) DL337
Magnified digit approximately .1"
Cathode for each digit
Segments are parallel for multiple operation
5-10 MA per segment
EACH $1.75 4 (12 DIGITS) $6.00)

RCA Numitron



EACH............$ 5.00

SPECIAL: 5 FOR $20.00

DR2010

MOS MEMORY 2102-2

1024 Bit Fully Decoded Static MOS Random Access Memory

- fast access 650ns
- fully TTL compatible
- n channel silicon gate
- single 5 volt supply
- tri-state output
- 1024 by 1 bit
- chip enable input
- no clocks or refreshing required

Brand New Factory Parts
16 PIN DIP Each $5.00
8 for $34.95

Power Supply SPECIAL!

723 DIP variable regulator chip 1-40V, + or - output @ 150 MA 10A with external pass transistor--with diagrams for many applications.
EACH $1.00 10 FOR $8.95

5001 Calculator

40-Pin calculator chip will add, subtract, multiply, and divide. 12-digit display and calculate. Chain calculations. True credit balance sign output. Automatic over-flow indication. Fixed decimal point at 1, 2, 3, or 4. Leading zero suppression. Complete data supplied with chip.

CHIP AND DATA........ONLY $2.49
DATA ONLY (Refundable)... $1.00
5002 LOW POWER CHIP AND DATA $12.95

High Quality PCB Mounting IC Sockets

8-PIN, 14-Pin, 16-Pin and 24-Pin PCB mounting ONLY--no wire wrap sockets.



8-Pin.............$ .22
14-Pin........$ .26
16-Pin........$ .30
24-Pin........$ .75
40-Pin..........$1.25

All IC's are new and fully tested. Leads are plated with gold or solder. Orders for $5.00 or more will be shipped prepaid Add $ .35 for handling and postage for smaller orders; residents of California add sales tax. IC orders are shipped within 2 workdays--kits are shipped within 10 days of receipt of order. $10.00 minimum on C.O.D.'s.

Mail Orders to: Phone
P.O. Box 41727 (916) 334-2161
Sacramento, CA
95841

BABYLON ELECTRONICS

Money back guarantee on all goods!

Dale Trimmer

- 12 turn trimpots which plug into a DIP socket
- 5K and 200K
- ¼" x ¼" x ¼"
- 4 leads spaced .3" x .2"

Each $1.00 10 for $8.95

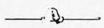
1000 MHz Counter

11C05 Fairchild 1GHz Divide By Four

- DC to 1000 MHz operation
- AC or DC coupled
- Voltage compensated
- TTL or ECL power supply
- 50 ohm drive output
- Lead compatible with Plessy SP613
- True and complement ECL outputs
- 14 pin DIP
- Data and application notes

Each $49.95

LED's



MV50 Red Emitting $ .20
10-4 MA @ 2V 10 FOR $1.25



MV5024 Red TO-18 $ .35
High Dome 10 FOR $2.95



MV10B Visible Red $ .30
5-7 MA @ 2V 10 FOR $2.50

CMOS

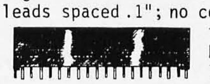
CD4001	$ .45	CD4023	$ .45
CD4002	.45	74C20	.65
CD4011	.45	74C160	3.25
CD4012	.45		

3-Amp Power Silicon Rectifiers

MARKED EPOXY AXIAL PACKAGE

PRV	PRICE	PRV	PRICE
100	$.10	800	$.30
200	.15	1000	.40
400	.18	1200	.50
600	.23	1500	.65

DIODE ARRAY 10-1N914 silicon signal diodes in one package. 20 leads spaced .1"; no common connections.

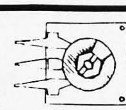


EACH.....$.29
10 FOR $2.50

7400 Series DIP

7400	.20	74H51	.25
74H00	.30	7453	.20
7401	.20	7454	.20
74H01	.25	74L54	.25
7402	.25	74L55	.25
7403	.25	7460	.16
7404	.25	74L71	.25
74H04	.30	7472	.40
7405	.30	74L72	.60
7406	.40	7473	.35
7408	.30	74L73	.75
74H08	.30	7474	.45
7410	.20	74H74	.75
7413	.75	7475	.80
7417	.40	7476	.55
7420	.20	74L78	.70
74L20	.30	7480	.50
74H20	.30	7483	.70
74H22	.30	7489	3.00
7430	.20	7490	1.00
74H30	.30	7492	.65
74L30	.30	7493	1.00
7440	.20	7495	.65
74H40	.30	74L95	1.00
7442	1.00	74107	.35
7447	1.50	74145	1.25
7450	.20	74180	1.00
74H50	.30	74193	1.50
7451	.20	74195	.65

25K Trimmer



PRINTED CIRCUIT BOARD TYPE
EACH $.20 10 FOR $1.50

Rectifiers



VARO FULL-WAVE BRIDGE

VS647	2A	600V	$1.10
MR810 Rectifier	50V	1A	$ .10

Special 811: Hex Inverter

TTL DIP Hex Inverter; pin interchangeable with SN 7404. Parts are brand new and branded Signetics and marked "811."

	EACH	$ .16
DATA	10 FOR	1.50
SHEET	100 FOR	14.00
SUPPLIED	1000 FOR	110.00



1 AMP RECTIFIER

1N4007 1KV PRV EACH $ .15
SALE 10 for $1.00



MAN 4



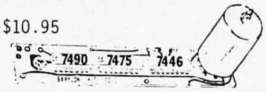
7-Segment, 0-9 plus letters. Right-hand decimal point. Snaps in 14-pin DIP socket or Molex. IC voltage requirements. Ideal for desk or pocket calculators!

EACH $1.20 10 OR MORE $1.00 EACH

CD-2 Counter Kit

This kit provides a highly sophisticated display section module for clocks, counters, or other numerical display needs. The unit is .8" wide and 4 3/8" long. A single 5-volt power source powers both the ICs and the display tube. It can attain typical count rates of up to 30 MHz and also has a lamp test, causing all 7 segments to light. Kit includes a 2-sided (with plated thru holes) fiberglass printed circuit board, a 7490, a 7475, a 7447, a DR2010 RCA Numitron display tube, complete instructions, and enough MOLEX pins for the ICs... NOTE: boards can be supplied in a single panel of up to 10 digits (with all interconnects); therefore, when ordering, please specify whether you want them in single panels or in one multiple digit board. Not specifying will result in shipping delay.

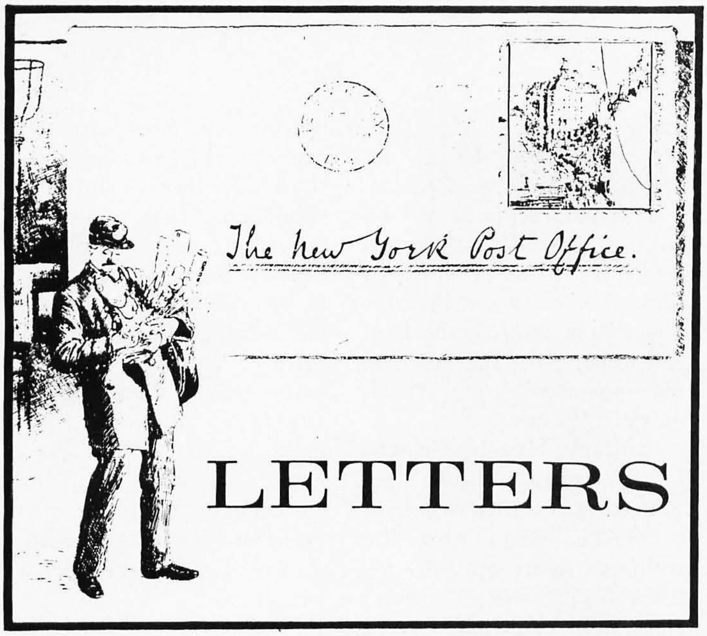
COMPLETE KIT ONLY $10.95
FULLY-ASSEMBLED UNIT $15.00



Boards supplied separately @ $2.50 per digit.

L I N E A R S

NE555	Precision timer	.90
NE560	Phase lock loop DIP	2.95
NE561	Phase lock loop DIP	3.00
NE565	Phase lock loop	2.95
NE566	Function generator TO-5	3.50
NE567	Tone decoder TO-5	3.50
709	Popular Op Amp DIP	.40
710	Voltage comparator DIP	.60
711	Dual comparator DIP	.45
723	Precision voltage regulator DIP	1.00
741	Op amp TO-5/MINI DIP	.45
748	Op Amp TO-5	.80
CA3018	2 Isolated transistors and a Darlington-connected transistor pair	1.00
CA3045	5 NPN transistor array	1.00
LM100	Positive DC regulator TO-5	1.00
LM105	Voltage regulator	1.25
LM302	Op Amp voltage follower TO-5	1.25
LM308	Op Amp TO-5	2.00
LM309H	5V 200 MA power supply TO-5	1.00
LM309K	5V 1A power supply module TO-3	1.00
LM311	Comparator Mini	1.75
LM370	AGC amplifier	1.75
LM380	2-Watt Audio Amp	1.75
LM1595	4-Quadrant multiplier	1.70
MC1536T	Op Amp	1.35



A TALE OF TWO HOBBIES

Enclosed is $10 to cover my charter subscription to BYTE. I read about the magazine in yesterday's issue of HR Report. I think that it is a great idea! It fills a need that has been created by the recent boom in the computer hobby area.

I am a ham radio operator (WB6ASR) and very interested in interfacing this hobby with computers. Since BYTE is going to be published by 73 it will be a perfect magazine for those who share my interest in these two hobbies. I hope to see many articles relating to this subject. First off I would like to see BYTE Magazine lead the fight with the FCC to allow ASCII to be used in amateur radio along with (or instead of) baudot. This is a basic step that has to be achieved in order to easily interface the two.

"Amateur radio is a natural for the computer hobbyist."

Amateur radio is a natural for the computer hobbyist. I can see a network of computers tied together by repeaters. The group I am affiliated with (AMT, W6AMT, Box 1, Montebello CA 90640) is very interested in just that. For any help or information BYTE may like along these lines, feel free to contact us.

I would also like BYTE to cover all areas of computers. Software: We should have a library of programs for free exchange among subscribers. Hardware: The more common computers that are in the hands of hobbyists should receive consideration [all DEC computers (including the new micro-processors and micro-computers such as PDP-8A and LSI-11), the 8080 family, Altair, etc]. Peripheral: Terminals – what are the best ones for the lowest cost such as Decwritter II, floppy discs, tape – Dectape, paper, cassette.

Gregory D. Campbell
Montebello CA

Thank you for your thoughts, Greg. We'll be trying hard to fill the bill for the home brew computer area and its interface to amateur radio – this new field of computers in the home is going to be big. We're in the same stage relative to computing that transportation was at the turn of the present century – thousands of experimenters working on the applications and engineering of products which can eventually be mass produced . . . just as no one could imagine the eventual impact of automotive technology, the next half centry in small computer applications should be just as interesting.

Why, in only that single area of amateur radio interfaces, the applications are wide and varied: digital remote control stations, the repeater networks you mention, digital station logs, automated ham rigs, packet switching communications nets which will expand upon the old amateur radio telegram network concept, program exchange frequencies for computer-hams to get together upon; ASCII RTTY communications with intelligent transmitter/receiver rigs to send messages along with error correcting codes.

. . . CARL

THE DEADLY GRAPEVINE STRIKES AGAIN

BYTE people:

Please find enclosed my personal check for $10 to cover a first year subscription to BYTE.

I would also like to make a few comments regarding your apparent " 'computer freak' only" editorial policy.

Amateur radio today is one of the highest technology avocations to be organized worldwide. It is perhaps *the* hobby with the most political clout as well.

Fraternal attitudes have traditionally led to cooperative technology development in amateur radio. The possibilities of time-sharing VHF-UHF repeaters, packet-switched worldwide traffic/data communication networks, or even the digital uses of the OSCAR satellites seem logically to preclude exclusion of amateur radio from BYTE.

Encourage hams to become hackers, and hackers to get their licenses. The whole is too much greater than the sum of its parts to divide them at birth.

While I am on my soapbox, there are some thoughts about what I would like to see from BYTE. How about contests between various game programs (bet my checker program can beat your checker program . . .)? Standardization on a global (meta) language for the descriptions of programs and algorithms? For fun it might be nice to have a mathematical games column

"Looking backward I'm kinda glad it didn't function properly when first plugged in. Otherwise, I would never have learned what certain circuits are doing, why they are doing it, and how they are doing it."

(like *Scientific American* but hobby oriented). Also, a punched tape service for the dissemination of programs and other data would be popular. As the only (big) publication in the field (hmmm . . .) I hope that there is made some form of guidelines for the storage, format and medium of information. You could have "BYTE my ass" T shirts, "BYTE by BYTE" programming aids and booklets, pocket guides for innumerable things. You have really stumbled onto a whole new hobby at its birth (adolescence?).

This letter has really tired me out. Thanks for BYTE.

George Henry Flammer III
Stanford CA

PS: I was supposed to mention here something about a "life" membership or subscription. So I will.

Am I going to hang myself in a grapevine? This is probably the first magazine ever to get editorial criticism before a single issue is printed! (I'll qualify that: it's my first magazine . . .) The points are well taken. Examine the first issue and you'll find a breadth of articles ranging from games to hardware – and we even have a $99 introductory special on "Life Time Subscription . . ."

. . . CARL

NUTS AND GUTS

Happy to hear about BYTE being initiated. If it is in the tradition of *73*, you got one perennial subscription. Since I'm not a ham buff, I only occasionally look at an issue of *73*, but am impressed with their content and format. They haven't been afraid to provide articles that cost more than 15 bucks and have a detailed discussion of the circuit which until recently has been my complaint with PE, RE, etc.

I remember back in '63 when I worked for the now defunct *Electronics Illustrated* magazine that the policy was tenth grade level and under $10, and how can we squeeze the articles between the ads.

Anyway, maybe in the near future, I can get something together to submit to BYTE. Right now I'm trying to figure out why my EXAM/DEP NEXT won't work right on Altair. I had a hell of a lot of bugs in the unit until I got rid of the Molex Soldercons. An aside, a technique for plated thru holes filled with solder: Drilling out is not necessary, which ruins the hole plating and requires soldering on both sides of the board. After removing the Soldercon pins, I used round toothpicks which I pushed thru the hole while holding the iron to the soldered side.

Philosophically, computer hobbyists must be nuts or have a lot of guts, if I'm any indication. Here I buy a sophisticated piece of equipment only knowing vaguely how logic works, with a 40-hr course in FORTRAN, a VOM and an ill-adjusted 15-year scope; yet, I plunge head-long into a vast unknown.

Which brings up another want in future articles or at least my preference. I seem to prefer seeing timing diagrams along with circuit descriptions. At least this is what I found out when I analyzed my basic Altair to de-bug it. The Altair manual assumes "it'll work" right off or "if not, give us a call." So, my resort was to take each circuit function and write myself a description about it with logic equations, timing diagrams and everything I could remember from a course three years ago about logic design (which I incidentally took as a hobby interest to learn how those infernal machines functioned). Looking backward I'm kinda glad it didn't function properly when first plugged in. Otherwise, I would never have learned what certain circuits are doing, why they are doing it, and how they are doing it. Now back to that damn EXAM/DEP NEXT circuit.

Bill Fuller
Grand Prairie TX

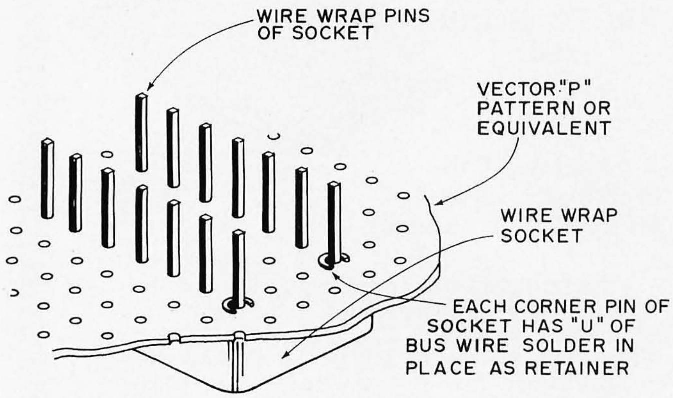
PS: Just got my copy of PCC May issue. Seems that others have had problems with the EXAM/DEP NEXT mono m/v in Altair. Anyway, with all my self-inflicted problems with the unit and some of theirs, I'm not disappointed in the unit considering the price of the 8080 chip was $360 at the time I went Altair; with their new price and the 8080 at $175 now, I might not go it. But as Baba Ram Dass says, "Do it now."

Thanks, Bill. Sounds like you have all the prerequisites needed for the home brew computer hobby – curiosity and initiative. The VOM and scope certainly help, as well as the course in FORTRAN. But the really important prerequisite is the desire to know how computing machinery and logic work. The art of computing is what BYTE is all about – translated into a personal use context. This art has come a long way since Napier's Bones, Babbage's engines, and Boole's formal logic . . . and it will go a long way in the future as well – through extensions of present technology and new ways of solving problems.

. . . CARL

Byter's Digest

A Quick Kluge for Fastening Wire Wrap Sockets to Perforated Board:



Using retainers to anchor wire wrap sockets.

If you want to wrap up a quick project, it is often handy to use perforated board (eg: Vector "P" pattern VECTORBOARD) to mount wire wrap sockets. I have used many methods for attaching boards — rivets, bolts, epoxy glue, cyanoacrylic glue, etc. One method which I dreamed up the other day to solve the mounting problem for a small test jig may prove useful to you at some point. Simply put the socket through the perforated board, then solder each corner pin with a "U" shaped retainer made of bus wire (e.g. about 14 to 18 gauge.) The result is a strong mechanical placement. When you solder on the retainer, use solder sparingly and employ a soldering iron with a narrow tip — of about 25 Watt capacity. Try to keep the solder as low on the pin as possible. When you wrap the circuit begin the first level of wraps higher up on the pins with retainers, to avoid the solder near the fastening.

CARL

A Note For Altair 8800 Users

A copy of an advertising sheet was sent to me by Gordon French of Menlo Park, California, describing a set of 8800-compatible interface and memory cards. The advertising sheet for Processor Technology Co., 2465 Fourth Street, Berkeley CA 94710, mentions the following items:

1. 4k Memory Board Kit with optional 1k ($85), 2k ($125) or 4k ($225) variations.
2. PROM Card Kit for Intel 1702A or National 5203 ultra-violet erasable PROM's, comes with address decode but not PROM's (you'll need a programmer and PROM chips) at $45.
3. I/O Board Kit providing both parallel and serial interfaces to the "real world." A UART is used for the serial interface optionally under program control, with selectable baud rates, choice of four EIA plus TTY and TTL serial interfaces ($125).

The ad sheet said delivery begins June 1. If you're interested, I suggest you write these people to find out further details.

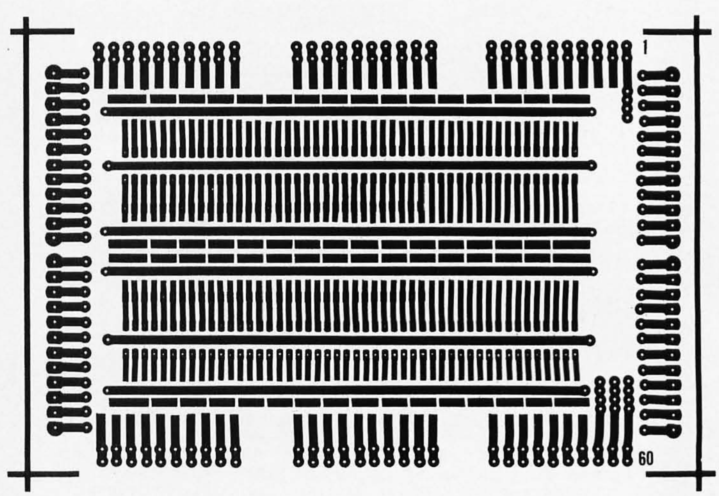
Electronic News (June 9, 1975) reports in an article by Paul Plansky the announcement of a 12-bit PDP-8 compatible CMOS processor chip to be produced by Intersil. This sounds like a great idea for the home brew computer market — but not for a while. The "hardware starter kit" is reported to cost $3050 for a set of three boards.

The boards include a memory board with 4k words, a CPU board with TTY interface, and a control panel board. The primary advantage outside of PDP-8 software compatibility is the CMOS nature of the product — the entire computer (all three boards) is quoted at a 2 mW requirement. (How this number is compatible with a TTY current loop output is not clear — but it is reasonable for the CMOS part alone.) This computer is not yet in a position where it can be used by the home brew computer market — but the idea of a PDP-8 (or PDP-11) compatible home brew machine is quite attractive due to the large amount of "public domain" software available for these machines.

James Fry's Prototyping Board

James Fry, PO Box 6585, Toledo OH 43612, sends along the layout of a general purpose prototyping board designed to mate with the connectors of the original TVT-1 TV Typewriter design of Don Lancaster. Jim has used his board in the process of modifying the design to mate with his 8008 system.

The holes at the edge of the board allow for additional input and output connections plus the mounting of the Molex pin and socket combinations needed to extend the stack of TVT-1 boards by an additional layer. Jim will provide copies of this board undrilled for $6 postpaid, or you can take the idea and lay out your own version for contact printing and home fabrication.

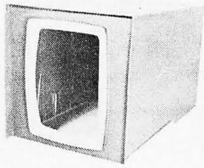


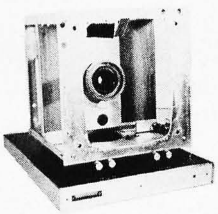
We've got a bunch of these fantastic video display terminals . . . and we've got a little problem. We promised Sanders Associates that we would sell them as scrap. A couple of wires disconnected makes them scrap, right? These VDTs should be great for SSTV, for a CW/RTTY keyer terminal, an oscilloscope, weather satellite monitor, or even a computer terminal (which they were). We've tested some of these and they seem to be near-perfect. You aren't likely to find a VDT system like this for less than ten times the price . . . so order several right away while we've got 'em.



ITEM G: ASCII KEYBOARD – This is the ASCII encoded keyboard used with the SANDER'S ASSOCIATES 720 System Terminal. Plugs into the front of the chassis mounting base. Makes a very professional Video Readout Terminal combination. These keyboards are in like new condition, have interconnection data etched on the IC-Diode matrix PC board. They can be readily used for any ASCII encoded requirement. Similar keyboards, when available, sell for almost two times the very low SUNTRONIX price of – $49.95. **PPD**

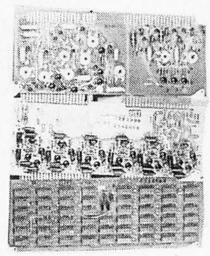
ITEM F: ENCLOSURE AND BEZEL FOR 12" CRT – This is the frosting on the cake. All components A thru E fit perfectly inside this enclosure. It is hinged and can be lifted for easy access to the electronics. It will really dress up any project. Measures approx. 22"L x 18"W x 20"H and weighs approx. 10 lbs. Made of steel with a handsome blue crackle finish. Get 'em while they last, for – $11.95 (incl. bezel) FOB.



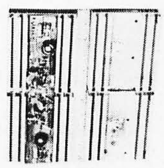


ITEM B: BASIC CHASSIS AND MOUNTING BASE for 12" big-screen CRT. Tube can be mounted either vertically or horizontally by rotating front plate 90 degrees. Comes with base, on-off sw. and intensity control, four controls for vert. and horiz. Has plenty of room for most any electronics needed for your pet project. All subassemblies offered will perfectly fit in spaces provided. Why try to cut the metal yourself? This chassis will let you concentrate on the electronics instead of the metal-work!! Order now for only – $14.95 FOB, less CRT.

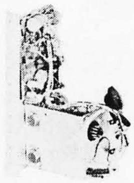
ITEM C: FOUR PC BOARDS CHOCK-FULL OF GOODIES – Two D/A converters, one IC-loaded logic board, and one multipurpose board. We have no schematic data for these boards at present. We will supply any data we obtain to purchasers as we get it. Of course when we finally figure out what these boards are good for, the price will change accordingly. Take the gamble now and we'll provide any data we get free of charge. Buy all four boards or just one – $1.50 ea. (our choice) or all four for $5.00. **PPD**

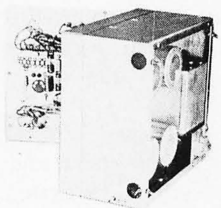


ITEM A: VERTICAL AND HORIZONTAL AMPLIFIER Subassemblies – Good for a conservative 150W complementary DC coupled output. Freq. resp. beyond 2.0 MHz. Parts alone worth many times the low, low price of – $6.95 ea., or both for $10.95 **PPD**



ITEM D: CRT HIGH VOLTAGE POWER SUPPLY – This is a real super CRT High Voltage Power supply, providing all voltages needed for any CRT. Outputs 10-14KV DC, plus 490 Vdc, minus 150 Vdc. Needs inputs of plus 5.0 VDC, plus 16.0 VDC and a drive signal of approx 8.4 kHz @ 1.0 vrms or more. All inputs/outputs via plug/jack cables and even has a socket/cable assy for the CRT. A very fine buy at only – $14.95 (incl. data) **FOB**





ITEM E: LOW VOLTAGE POWER SUPPLY – A real brute used to supply all low voltages needed by the original 720 CRT Terminal. Input, 117VAC, outputs: plus 16.0 VDC @ 10.0 A; minus 16.0 VDC @ 10.0A; plus 5.0VDC @ more than 2.0A, all regulated. Mounts on the rear of the Basic Chassis (Item B) Weighs approx 45 lbs and will be shipped with interconnection data for only – $19.95 FOB.

PACKAGE DEAL – For the really serious experimenter we'll make a very special offer – you can buy all of the sub-assemblies listed above plus a good 12" CRT, a muffin fan for cooling. We'll supply instructions for interconnection for all subassemblies so that you can, within minutes after receiving this once-in-a-lifetime deal, put an X-Y display on the CRT. We'll also include a list of possible applications for those with short imaginations! Don't miss out on this real money-saving buy; the individual prices for the sub-assemblies add up to $127.70. You can buy the entire package for a very low package price of – $79.95 FOB.

On all postpaid orders, please ADD $1.50 to cover handling costs. Orders shipped same day in most cases.

master charge THE INTERBANK CARD

BANKAMERICARD

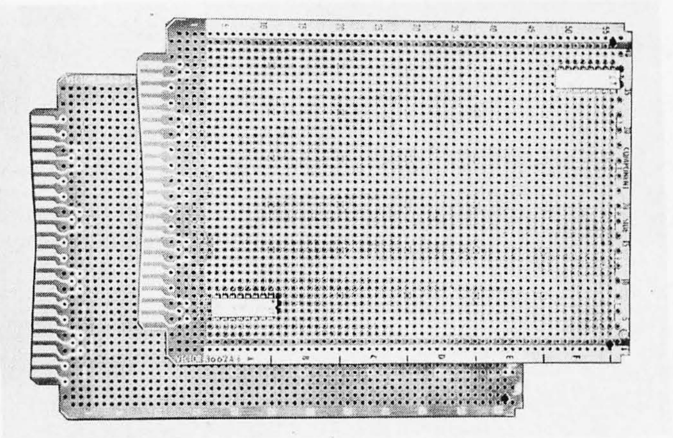
SUNTRONIX company

6 KING RICHARD DRIVE, LONDONDERRY, N. H. 03053

603-434-4644

Byter's Digest

The Inventors of the Prepunched Perforated Board Have Done it Again.



Edge connector configuration on Vector 3662A-6 Plugbord reduces insertion forces by 2-1/2 lbs.

The Vector Electronic Co., Inc., announces a new variant on a traditional theme – the idea of a printed circuit edge connector with a lower insertion force due to a unique design.

Special W-shaped cut on the printed circuit board edge connector reduces insertion forces by two to eight pounds, depending on the type of receptacle and the number of contacts. First used on Vector Electronic Company's new Model 3662A-6 board, the "W" cut consists of a slight inwardly tapering chamfer across the face of the card edge which allows the board to be inserted with a gradually increasing force instead of the conventional high peak pressures. In addition, the two outside ground terminals engage first and disengage last to protect circuits if the board is inserted or removed with the power on.

According to the press release this technique will be used on all new Plugbords manufactured by Vector.

Vector cites tests on 22/44 edge connectors with terminals on 0.1 in. centers which indicate that average insertion force with the "W" cut is 9 lbs. compared to 11 lbs. with a conventional straight configuration. Average force on 15/30 connectors with 0.156 in. centers drops from 13 lbs. to 6.5 lbs. Similarly, average insertion force on 22/44 connectors with 0.156 in. spacing is reduced from 16.5 lbs. to 14 lbs., while the pressure on 36/72 connectors with 0.10 in. spacing declines from 31 lbs. to 23 lbs. Withdrawal forces are unchanged by the new technique.

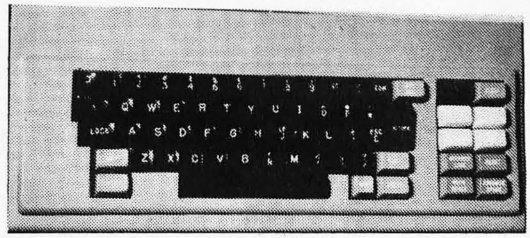
The Vector Model 3662A-6 board illustrated has 22/44 edge contacts with 0.156 in. spacing. Contacts are two-oz. copper, nickel plated and gold flashed for long life and low contact resistance. Individual contacts are numbered for easy identification. The 4.5 in. by 6.5 in. by 0.0625 in. board uses a new blue-colored FR4-type epoxy laminate which meets or exceeds MIL-P-13939-E GF. Prepunched 0.042 in. dia. holes are spaced on 0.1 in. centers. The top surface of the board has markers for the number one pin of 14-pin DIPs across the field and component placement indices around the board's periphery. The reverse side has a two-oz. copper ground plane with solder coating.

The boards are priced at $7.55 in unit quantities with volume discounts available. Delivery is from factory stock.

Vector Electronic Co., Inc., 12460 Gladstone Ave., Sylmar CA 91342. 1-213-365-9661; TWX 910-496-1539.

THE BIT BUCKET is the name of a new publication now available from National Semiconductor Corporation (mailing address: COMPUTE/470, National Semiconductor Corporation, 2900 Semiconductor Drive, Santa Clara CA 95051). *The Bit Bucket* is a user "newsletter" supported by National, with a subscription tab of $15 per annum, for the purpose of exchanging info on the products National manufactures: IMPs of all kinds and the new PACE 16-bit micro. The info I got (in Volume One, Number One) includes seminar schedules, plugs for products of members of the club, a solicitation of members in COMPUTE (Club of Microprocessor Programmers, Users and Technical Experts), descriptions of an assembler and listings of the library of complimentary packages of software. If your home brew system idea is growing aPACE, you might find this publication useful ... **CARL**

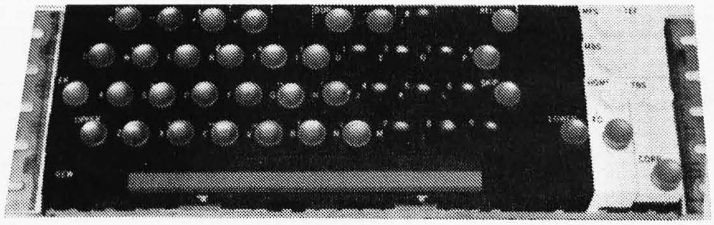
RAYTHEON KEYBOARDS



From Raytheon Corp. and we were told they were made for the FAA in air traffic control computers. Appear to be unused condition. Switches are magnetic reed relay. Ascii encoded with PC board mounted under the switch board. Sorry to say but have no data with these at this time.

Ship wgt. 6 lbs. . . . . #SP-139 $30.00

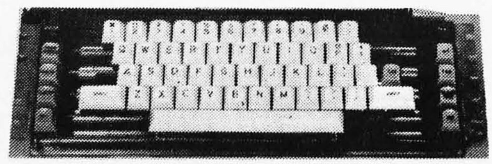
HONEYWELL KEYBOARD



A nice purchase from Honeywell of these unused keyboards with reed relay, magnetic switching. No encoder with these and they can be used in a variety of ways . . . Morse code generator, TV print out, terminal keyboard for computer work, etc. If you want an all purpose keyboard in new condition, this is it. Only about a hundred left. The price is deserving of a second look as it's a give-away.

Ship wgt. 5 lbs. . . . . #SP-165 $20.00

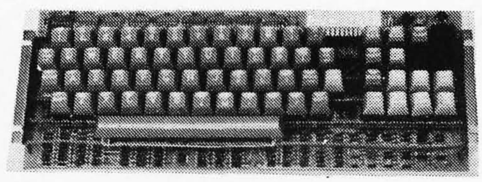
ASCII ENCODED



From one of Americas largest manufacturers of keyboards. ASCII encoder mounted beneath board using ICs Picture shown is typical keyboard. There seems to be no end to customers wanting keyboards and we are lucky enough to keep coming up with more. These are clean and with all keytops in place.

Ship wgt 6 lbs. . . . . #SP-122 $35.00

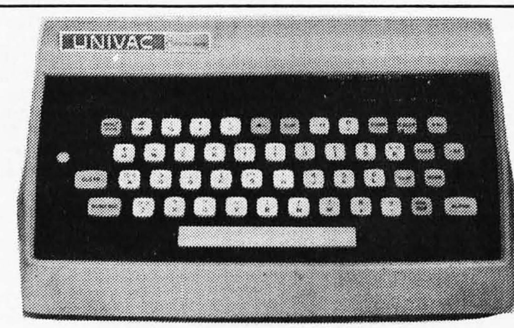
KEYBOARDS



A bargain in computer keyboards with encoder board attached. The low price is due to the fact that a few keytops may be missing. So if you can improvise, you acquire a bargain keyboard. We will furnish missing keytops though they may not have the correct letter, but you can stick any letter you want with tape. At $10 you can hardly miss.

Ship wgt. 6 lbs. . . . . #SP-123 $10.00

UNIVAC KEYBOARD



This keyboard with encoder (Holarith) mounted in dust proof enclosure has been one of our best movers no doubt due to its handsome desk top appearance. The encoder board is easily removed for re-working and the case is gray plastic. We have sold over a thousand of these to date and were sold out. But along came a supplier with several hundred more and we are back in business. Same price as we sold them for the last two years, no upgrading of price.

Ship wgt. 6 lb. . . . . #SP-124 $35.00

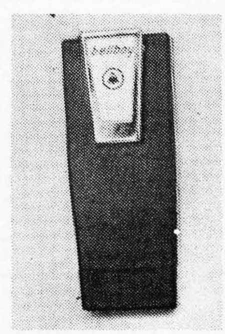
Please add shipping cost on above.

MESHNA PO Bx 62 E. Lynn Mass. 01904

Meshna

FREE CATALOG

BELLTONE PAGER



Genuine "Ma Bell" belt clip radio receiver beeper. Picks up specific radio signals in 35 MHz area, encoded by internal reed encoder. Seems to be a "natural" for construction jobs, in-plant calling. An interesting experimental gadget. Self contained antenna, adjustable coding by shifting wires on coding module.

#SP-125 $5.00 each, 6/$25.00

THE COMPUTER "SYSTEM" CONCEPT

A COMPUTER with a BUILT-IN CONSOLE TERMINAL from SPHERE

The SPHERE 1 computer system was designed to provide an uncompromising computer system at minimal cost.

Keyword... "System"

The keyword to our design is the world "SYSTEM". Every phase of the design has been influenced by the "SYSTEM" philosophy. To justify the system title, a "COMPUTER" must perform an application acceptably. Recently the cost of peripherals and software have substantially exceeded the cost of the computer, but without them, a computer cannot perform much of anything acceptably.

With the onset of the micro-processor, real design innovations have been possible, but without the system philosophy, a micro-processor can only reduce the processor cost. Peripherals, memory, and software continue to be expensive.

The SPHERE 1 computer is uniquely cost effective because it utilizes real design innovations to reduce the amount of circuitry required throughout the system. The SPHERE add-on memory board will support 4, 8, 12, or 16K of dynamic random access memory (instead of four 4K memory boards and a mother board). Our power supply has been placed in a separate chassis to eliminate a common source of heat. This allows the system to run cooler and eliminates the need for an expensive fan. The system uses a standard TV for a 512 character display. The use of the TV and other common components has reduced the cost and allowed more machine versatility. Further cost reductions have been ächieved by replacing the front console (lights and switches) with the TV terminal and a program in Read Only Memory (ROM) that performs the same function, only better.

The Processor... A One Card Control System

The CPU card is packaged to provide all of the basic functions required by a useful system. It contains a Motorola M6800 micro-processor which is the most advanced micro-processor on the market today. The CPU Module also includes 4K words of random access memory which is the "minimum" required to perform useful functions. Sixteen lines of digital I/O have been provided as on option on this board. This allows the module to act as a stand-alone "system" in many instances. Further innovations have been added to enhance it's "system" capability. They are:

1)a Real-Time clock with INTERUPT capability at 31, 62, 125, 250, & 500 HZ.

2)buss lines are "high-drive" buffered to run many more peripherals.

3)the system buss is driven over flat-cables which means no mother-board is required for expansion and the system may be configured with space utilization efficiency.

4)the CPU has been provided with 1K of Programmable Read-Only memory. This memory can contain a complete process control program for many applications. When initially delivered, it contains the PDS system which is described later.

Peripherals... Floppy Disks, Line Printers, Paper Tape, Terminals...

In order to insure a full offering of high quality peripherals from the onset, we have selected manufacturers who already have peripherals which interface to our product. This philosophy has allowed us, in the case of our disk, to select already running software (namely a disk operating system) which we may offer to our users immediately. Other peripherals that are available with our system include a low cost line printer and a paper tape reader/perforator. These devices are interfaced to the system via a single interface module which also serves as a programmable digital Input/Output port.

The Keyboard module includes tactile feed keyswitches, 2 key roll-over encoding, a numeric keypad and a star cursor editing keypad. The SPHERE system also supports the lowest cost terminals available today.

PDS.... unparalleled

The Program Development System (PDS) includes an EDITOR, and ASSEMBLER, and a debugging package. It also includes CRT display and audio cassette software drivers, plus a cassette loader and dumper. Although most computer processing occurs at the character (8 BIT) level, it is sometimes desirable to use 16 bit arithmetic so we have provided an extended instruction set in the PDS system. The extended instructions include 16 bit multiply, divide, add, subtract, etc. The instructions include input/output and binary (16 bit) to ASC11 to binary conversion. PDS is entirely contained in the read-only memory of the CPU module. It rounds out the "SYSTEM" concept of our smallest systems.

Basic Language FREE!!

The BASIC package includes t following utility commands: APPEN CATALOG, DELETE, GET, KIL LENGTH, LIBRARY, LIST, NAM RENUMBER, RUN, SAVE, AN SCRATCH.

The operators are: =, less tha greater than, less or equal, greater equal, not equal, AND, OR, NOT, MA MIN. The statements are: CHAIN, CO MON, DATA, Dim, EN FOR...NEXT, GO TO, TO...O GOSUB, IF...THEN, IMAGE, INPU LET, NEXT, PRINT, PRINT USIN READ, REM, RETURN, STOP. T functions are DEF, ABS, EXP, IN LOG, RND, SQR, SIN, TAN ATN, LE SGN, TAB. Matrix operations are: DO MAT IDN, MAT ZER, MAT CON, MA INPUT, MAT PRINT, MAT +, MAT MAT *, =, MAT TRN, MAT INV. F processing statements are: OPEN, KIL FILES, PRINT # READ #, END #. F string processing is supported.

This package will run in a 20 system with about 8K for user program An 8K subset of our BASIC is availab with 4K available for user programs. A Sphere software is a part of th "SYSTEM" price, and is available "SYSTEM" users for a minimal copyi fee.

The FLOPPY DISK OPERATIN SYSTEM (FDOS) is supplied on a systems purchased with a disk un FDOS is an extended PROGRA DEVELOPMENT SYSTEM. It provid for named files, an extended editor, full assembler, and debugging syste This system includes a comprehensi 300 page programming manual.

System Concept a Commitment

The software supplied to make Sphere System a useful "SYSTEM" attractive; however, the real contributi that SPHERE offers is one commitment. The SPHERE "SYSTE concept demonstrates only the surf of the real technological advances t are possible when true design innovati is combined with foresight a state-of-the-art technology. T SPHERE "SYSTEM" concept is t commitment.

WATCH AND SEE.

OEM'S CHECK WITH US...
WE'VE GOT COMPLETE ONE BOA CONTROL SYSTEMS FROM UND $600

AMBITIOUS TYPES CHECK WITH US
WE'VE GOT COMPLETE SYSTE STARTING AT $650. (KITS)

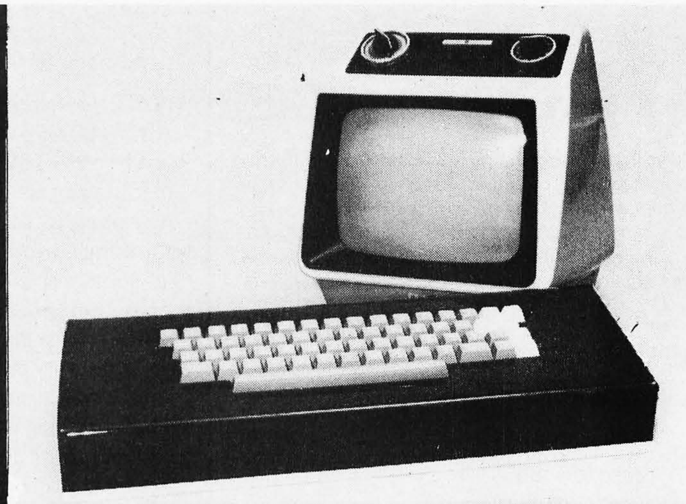
Bank Americard and MasterCharge accepted.

SPHERE CO

96 E. 5th South, Bountiful, Utah 840

$650 HOBBIEST!

- 8-BIT PARALLEL COMPUTER
- 4K WORDS of read/write memory
- MOTOROLA 6800 MICROPROCESSOR
- KEYBOARD WITH NUMERIC KEYPAD





$750 INTELLIGENT!!

USER PROGRAMMABLE

- FIRMWARE ASSEMBLER, EDITOR LOADER & EXTENDED INSTRUCTIONS
- 16 LINE X 32 CHARACTER DISPLAY
- 23 KEY KEYBOARD
- BUILT IN MODEM
- AUDIO CASSETTE INTERFACE

CASSETTES AND TV'S SHOWN FOR ILLUSTRATION ONLY

$1345 BASIC

- FULL EXTENDED BASIC
- 20K BYTES
- 512 CHARACTER CRT
- KEYBOARD WITH NUMERIC KEYPAD
- MODEM & AUDIO CASSETTE INTERFACE

OPTIONAL DESKTOP DISPLAY & VIDEO MONITOR EXTRA



LOWEST COST "SYSTEMS"

INTRODUCTORY OFFER ENDS SEPTEMBER 30, 1975

SPHERE

96 EAST 500 SOUTH - BOUNTIFUL, UTAH - 84010

BYTE reader service

To get further information on the products advertised in this issue of BYTE merely tear, rip, or snip out this advertiser index, fill out the data at the bottom of the page, mark the appropriate boxes, and send the works to BYTE, Peterborough NH 03458. Readers get extra Brownie Points for sending for information since this encourages advertisers to keep using BYTE — which in turn brings you a bigger BYTE.

ADVERTISER INDEX

To help the editors with a profile of the readers — what type of work do you do?

Have you a microprocessor running yet? and which, if so?

Messages for the editor:

Reader's Service
BYTE
Green Publishing Inc.
Peterborough NH 03458

SEPTEMBER 1975

BYTE acquired via
☐ Subscription
☐ Newsstand
☐ Stolen

Please print or type.

Name ______________________

Address ______________________

City ______________ State ______ Zip ______

Coupon expires in 60 days . . .

How BYTE started

from page 9

BYTE — make it a 24 pager. After talking the idea over with a couple of the manufacturers in the field it was obvious that we had been thinking too small. Okay, let's make it 5000 copies. The first announcement of the project was made in Hotline, an amateur radio newsletter with a very small circulation. The reaction was immediate: subscriptions began to come in at a good clip.

As mailing lists came in from manufacturers and as the word spread, the first issue print run was upped to 10,000 . . . then 25,000 . . . 35,000 . . . and finally 50,000 copies! As promises of ads came in there was a scramble to get enough articles to keep up with the ads. Ads are certainly of interest, but we didn't want to publish an all advertising magazine.

No apologies are needed for the articles in this first issue — between Carl's contacts and mine we got things started. It would have been a lot easier if our original idea of a 1000 copy 24 page magazine (with maybe 30% ads) had come about. On the other hand, here is a great opportunity for all of you readers to get busy at your typewriter and pass along your particular area of expertise. The need for good articles is great . . . material for the rank beginners as well as the sophisticated computer designers . . . hardware . . . software . . . surplus conversions . . . applications.

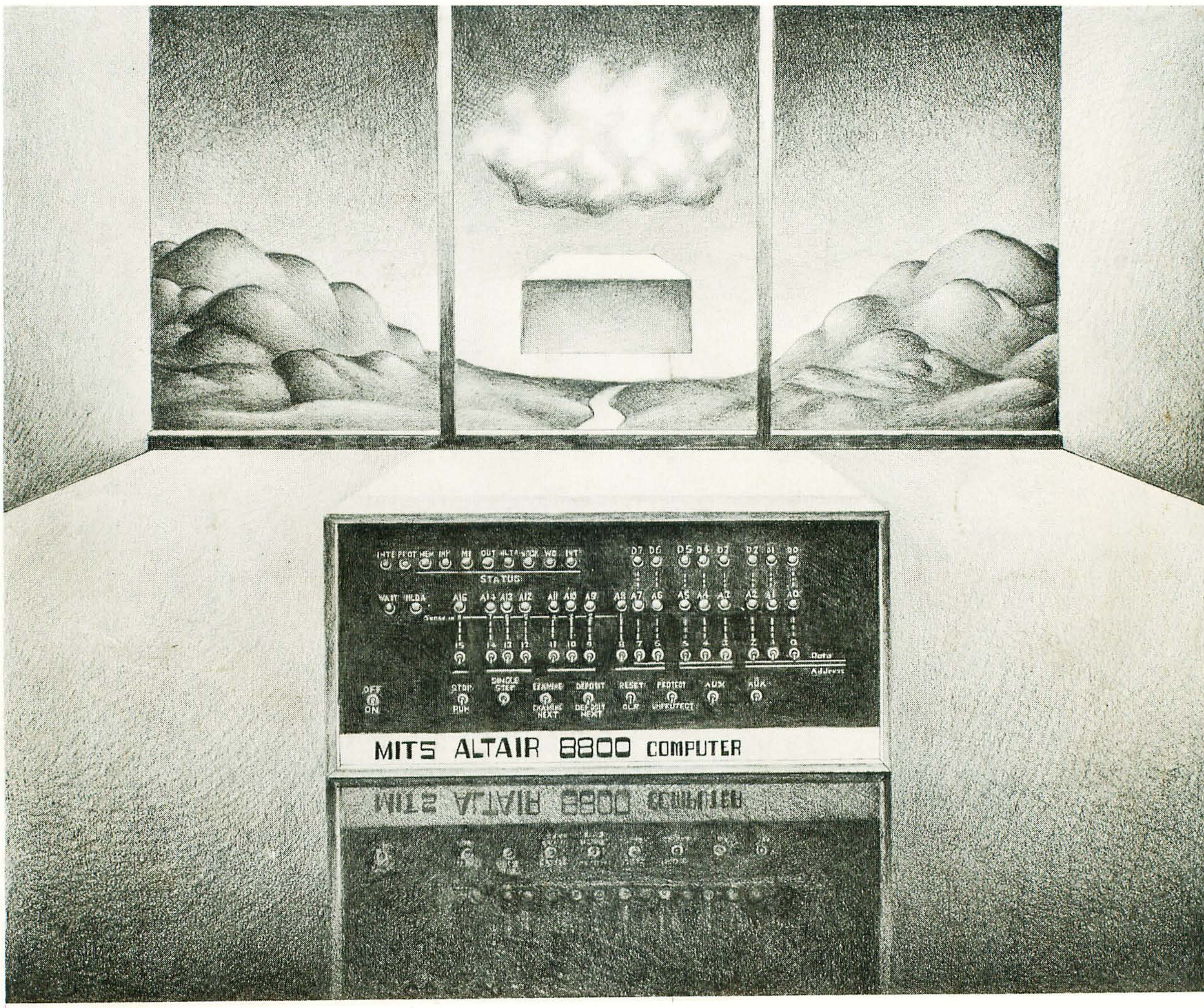
As we build a body of hobbyists, the market for reasonably priced equipment will be almost inexhaustible . . . microprocessors, video display units, keyboards, tape gear, discs, teletypes . . . endless list. MITS, RGS, Scelbi and Southwest Tech have a good start . . . are you going to let them make all the money?

Speaking of MITS et al, it didn't take me long to get one of the Altair 8800s to see what I could do with it. I'm afraid I didn't make it very far into the instruction book. I've got some more memory coming for it as well as their extended basic program and some I/O interfaces to hook onto a teletype or a VDT. I do have a VDT unit up and working . . . the Southwest Technical job which we got in kit form and which was assembled over a weekend on a card table, with a good deal of the work being done by my 12 year old daughter. And, believe it or not, the unit works! We all agree that it was a lot of fun to assemble and we're glad we went the kit route . . . we wouldn't have missed the fun. SWTPC sure did a fantastic job of getting that kit designed and produced.

Well, that's how BYTE got started. Now it's up to you . . . you can guide the magazine with your advice . . . with your articles . . . and with your support in getting more subscribers. We'll do all we can to make the magazine accurate, have plenty of interesting ads, look nice and come out on time. None of this is easy, of course, but we're in one of the nicest areas in the country — in southern New Hampshire — working in a 220 year old colonial mansion — and we have an efficient system where everything except printing and mailing of the magazine is done under the one roof. If you happen to find yourself wandering around a bit northwest of Boston, why please drop in and say hello. We're very friendly and the atmosphere is unbelievably relaxed . . . except near press time.

. . . WAYNE GREEN

Created by Man.



The Affordable Computer.